ДОДАТОК А

Код агента клієнт

```
using System;
using System.Collections;
using System.Collections.Generic;
using DG.Tweening;
using Pathfinding;
using UnityEngine;
using UnityEngine.UI;
using Utils;
using Random = UnityEngine.Random;

[RequireComponent(typeof(AIPath))]
public class Customer : MonoBehaviour
{
    public event Action<Customer, MagazineManager>
CustomerRemoveEvent;
    public List<WantedResource> CustomerOrder => _order;

    [SerializeField] private Renderer customerBodyRenderer;
    [SerializeField] private Transform headObjectHolder;
    [SerializeField] private Transform resourcesHolder;
    [SerializeField] private Animator customerAnimator;
    [SerializeField] private float checkDelay = 0.2f;
    [SerializeField] private Image progressImage;

    private MaterialPropertyBlock _materialProperty;
    private readonly int Color1 =
Shader.PropertyToID("_Color");

    private Vector3 _beginPosition;
    private AIPath _aiPath;
    private List<WantedResource> _order = new
List<WantedResource>();
    private List<CollectableObject> _orderedObjects = new
List<CollectableObject>();
    private MagazineManager _magazineManager;

    public int CurrentItemsCount = 0;

    private int _currentOrder = 0;
    private int _currentOrderItemCount = 0;
    private bool _wantWrapping = true;
    private bool _targetReachedDestination = false;
    private Coroutine curentCheck;
    private CustomerState _currentState =
CustomerState.GoToTarget;
    private readonly int Movement =
Animator.StringToHash("Movement");
```

```
    private readonly int Holder =
Animator.StringToHash("ObjectsInHands");
    private Coroutine fillCoroutine;
    private TargetOrderUI ui;

    public void SetupCustomer(Vector3 position,
List<WantedResource> order, MagazineManager manager)
    {
        position.y = transform.position.y;
        _beginPosition = position;
        transform.position = position;

        _order = order;
        _aiPath = GetComponent<AIPath>();
        _magazineManager = manager;

        _currentOrder = 0;

        SetCustomerDestination();
        curentCheck = StartCoroutine(CheckCustomersState());

        ui = MainInfoUI.Instance.GetOrderUI();
        ui.Initialize(_order[_currentOrder], transform);
    }

    public void SetupCustomer(Vector3 position,
List<WantedResource> order, Color bodyColor, MagazineManager
manager)
    {
        SetupCustomer(position, order, manager);
        ChangeBodyColor(bodyColor);
    }

    public void SetupCustomer(Vector3 position,
List<WantedResource> order, GameObject head, MagazineManager
manager)
    {
        SetupCustomer(position, order, manager);
        ChangeHead(head);
    }

    public void SetupCustomer(Vector3 position,
List<WantedResource> order, Color bodyColor, GameObject head,
MagazineManager manager)
    {
        SetupCustomer(position, order, manager);
        ChangeBodyColor(bodyColor);
        ChangeHead(head);
    }

    public void GoToExit()
    {
        StartCoroutine(CheckCustomerExitState());
```

```
    }

    public void GoToCassa()
    {
        _wantWrapping = false;
        SetCustomerDestination();
        _currentState = CustomerState.Wrapping;
        StartCoroutine(CheckCustomerWrappingState());
    }

    public void FillProgress(float fillAmount)
    {
        fillCoroutine = StartCoroutine(FillImage(fillAmount));
    }

    public void StopFillProgress()
    {
        if (fillCoroutine != null)
        {
            StopCoroutine(fillCoroutine);
            fillCoroutine = null;
        }

        progressImage.fillAmount = 0;
    }

    private IEnumerator FillImage(float fillAmount)
    {
        while (progressImage.fillAmount < 1)
        {
            progressImage.fillAmount += fillAmount;
            yield return null;
        }
        yield return null;
    }

    public CollectableObject GetOrderedResource()
    {
        var resource = _orderedObjects[_orderedObjects.Count -
1];
        _orderedObjects.Remove(resource);
        return resource;
    }

    public void SetBouquetObject(Bouquet bouquet, bool useBox)
    {
        bouquet.transform.SetParent(resourcesHolder);
        bouquet.transform.DOMove(resourcesHolder.position,
0.3f);
    }

    public void CheckForWrapping(float wrappingChance)
    {
```

```
        var chance = Random.Range(0f, 1f);
        _wantWrapping = chance < wrappingChance;
    }

    private void ChangeBodyColor(Color bodyColor)
    {
        _materialProperty = new MaterialPropertyBlock();

customerBodyRenderer.GetPropertyBlock(_materialProperty);
        _materialProperty.SetColor(Color1, bodyColor);

customerBodyRenderer.SetPropertyBlock(_materialProperty);
    }

    private void ChangeHead(GameObject headObject)
    {
        Instantiate(headObject, headObjectHolder);
    }

    private void SetCustomerDestination()
    {
        var target = GetTargetForOrder();

        if (target == null) { return; }

        _aiPath.canMove = true;
        _aiPath.SetPath(null);
        _aiPath.destination = target.position;
        customerAnimator.SetBool(Movement, true);
        _currentState = CustomerState.GoToTarget;
    }

    private Transform GetTargetForOrder()
    {
        if (_currentOrder < _order.Count)
        {
            var resource = _order[_currentOrder].type;
            return
Magazines.Instance.GetCustomerTargetByResource(resource);
        }

        if (_wantWrapping)
        {
            ui.EnableWrapp();
            return
_magazineManager.Wrapper.GetCustomerTarget();
        }

        ui.EnableCassa();
        return _magazineManager.Cassa.GetCustomerTarget();
    }

    private IEnumerator CheckCustomersState()
```

```
    {
        while (_currentState != CustomerState.WaitingForBuy)
        {
            switch (_currentState)
            {
                case CustomerState.GoToTarget:
                    CheckTargetPlace();
                    break;
                case CustomerState.TakeProducts:
                    CheckResourceCollection();
                    break;
            }

            yield return new WaitForSeconds(checkDelay);
        }

        yield return null;
    }

    private IEnumerator CheckCustomerSellingState()
    {
        while (_currentState == CustomerState.WaitingForBuy)
        {
            if (_aiPath.reachedEndOfPath &&
!_aiPath.pathPending)
            {
                StopMoving();
                if (_wantWrapping)
                {

_magazineManager.Wrapper.AddCustomerToQueue(this);
                }
                else
                {

_magazineManager.Cassa.AddCustomerToQueue(this);
                }
                _currentState = CustomerState.GoToExit;
            }
            yield return new WaitForSeconds(checkDelay);
        }
        yield return null;
    }

    private IEnumerator CheckCustomerWrappingState()
    {
        _aiPath.SetPath(null);
        _aiPath.destination =
_magazineManager.Cassa.GetCustomerTarget().position;
        ui.EnableCassa();
        while (_currentState == CustomerState.Wrapping)
        {
```

```
            if (_aiPath.reachedEndOfPath &&
!_aiPath.pathPending)
            {
                StopMoving();

_magazineManager.Cassa.AddCustomerToQueue(this);
                _currentState = CustomerState.GoToExit;
            }
            yield return new WaitForSeconds(checkDelay);
        }
        yield return null;
    }

    private IEnumerator CheckCustomerExitState()
    {
        ui.DisableAll();
        _currentState = CustomerState.GoToExit;
        _aiPath.canMove = true;
        _aiPath.SetPath(null);
        var destination = _beginPosition;
        destination.x -= 3f;
        _aiPath.destination = destination;
        customerAnimator.SetBool(Movement, true);

        while (_currentState == CustomerState.GoToExit)
        {
            if (_aiPath.reachedEndOfPath &&
!_aiPath.pathPending)
            {
                StopMoving();
                Destroy(ui.gameObject);
                CustomerRemoveEvent?.Invoke(this,
_magazineManager);
                Destroy(gameObject);
            }
            yield return new WaitForSeconds(checkDelay);
        }
        yield return null;
    }

    private void CheckTargetPlace()
    {
        if (_aiPath.reachedEndOfPath && !_aiPath.pathPending)
        {
            StopMoving();
            _currentState = CustomerState.TakeProducts;
        }
    }

    private void StopMoving()
    {
        _aiPath.canMove = false;
        customerAnimator.SetBool(Movement, false);
```

```
    }

    private void CheckResourceCollection()
    {
        if (_currentOrder >= _order.Count && _currentState !=
CustomerState.WaitingForBuy)
        {
            SetCustomerDestination();
            _currentState = CustomerState.WaitingForBuy;
            StopCoroutine(curentCheck);
            StartCoroutine(CheckCustomerSellingState());

            return;
        }

        if (_currentOrderItemCount >=
_order[_currentOrder].count)
        {
            _order[_currentOrder].orderCompleted = true;
            _currentOrder += 1;
            _currentOrderItemCount = 0;

            SetCustomerDestination();

            _currentState = CustomerState.GoToTarget;
        }
        else
        {
            var magazine =
Magazines.Instance.GetMagazineByType(_order[_currentOrder].typ
e);
            var resource = magazine.GiveResource();

            if (resource != null)
            {
                resource.transform.SetParent(resourcesHolder);

resource.SetGlobalPosition(resourcesHolder.position);
                resource.RotateToBaseValue();
                _orderedObjects.Add(resource);

                _currentOrderItemCount += 1;
                CurrentItemsCount += 1;
                _order[_currentOrder].currentCount =
_currentOrderItemCount;
                customerAnimator.SetBool(Holder, true);
            }
        }
        var currentOder = Mathf.Clamp(_currentOrder, 0,
_order.Count - 1);
        ui.ActualizeRow(_order[currentOder]);
    }
}
```

ДОДАТОК Б

Код агента співробітника

```
using System;
using System.Collections;
using System.Collections.Generic;
using DG.Tweening;
using Pathfinding;
using Sirenix.OdinInspector;
using UnityEngine;
using Utils;

[RequireComponent(typeof(CharacterData))]
public class WorkerBehaviour : MonoBehaviour
{
    [SerializeField] private WorkerType workerType =
WorkerType.Casier;
    [SerializeField] private Animator workerAnimator;
    [SerializeField] private Vector3 onTargetRotation;
    [SerializeField] private ObjectsHolder objectsHolder;
    [SerializeField] private PompObject pomp;
    [SerializeField] private Transform waitingPosition;
    [SerializeField] private ObjectSettings resourceToTarget;

    private readonly int Movement =
Animator.StringToHash("Movement");
    private readonly int Holder =
Animator.StringToHash("ObjectsInHands");

    private Transform currentTarget = null;
    private AIPath _aiPath;
    private CharacterData _data;
    public CharacterData Data
    {
        get
        {
            if (_data == null)
            {
                _data = GetComponent<CharacterData>();
            }

            return _data;
        }
    }

    private WorkerState _workerState;
    private ObjectGenerator _honeyZone;
    private ExportMachineObject _exportMachine;
    private ObjectCreatorMachine _objectCreatorMachine;
    private MagazineManager _manager;
```

```
    public void Initialize(MagazineManager magazineManager)
    {
        _aiPath = GetComponent<AIPath>();
        _data = GetComponent<CharacterData>();
        _honeyZone = Magazines.Instance.GetHoneyZone();
        _exportMachine =
Magazines.Instance.GetExportMachine();
        _manager = magazineManager;

        objectsHolder.SetCharacterData(_data);

        FindWorkerTarget();
        if (workerType == WorkerType.Placer || workerType ==
WorkerType.CottonPlacer)
        {
            pomp.SetGlobalWorker(this);
        }
    }

    public bool MatchType(WorkerType _type)
    {
        return _type == workerType;
    }

    private void FindWorkerTarget()
    {
        switch (workerType)
        {
            case WorkerType.Placer:
                FindPlacerTarget();
                break;
            case WorkerType.CottonPlacer:
                FindPlacerTarget();
                break;
            case WorkerType.Casier:
                currentTarget = _manager.Cassa.workerTarget;
                break;
            case WorkerType.Filler:
                FindFillerTarget();
                break;
            case WorkerType.CottonFiller:
                FindFillerTarget();
                break;
            case WorkerType.Wrapper:
                currentTarget = _manager.Wrapper.workerTarget;
                break;
            default:
                throw new ArgumentOutOfRangeException();
        }

        SetTargetDestination(currentTarget.position);
    }
```

```
    private void FindPlacerTarget()
    {
        if (objectsHolder.ObjectsCountInHand() > 0 &&
_workerState == WorkerState.FillVitrine)
        {
            if (resourceToTarget != null)
            {
                if (_objectCreatorMachine == null)
                {
                    _objectCreatorMachine =
Magazines.Instance.GetObjectByResource(resourceToTarget);
                }

                currentTarget =
_objectCreatorMachine.workerTarget;
            }
            else
            {
                currentTarget =
Magazines.Instance.GetWorkerTargetByResource(objectsHolder.Get
LastObjectSettings());
            }

            return;
        }

        if (objectsHolder.HandsIsEmpty() && _workerState !=
WorkerState.CollectObject)
        {
            _workerState = WorkerState.Watering;

            if (pomp.Automated)
            {
                _workerState = WorkerState.CollectObject;
                FindWorkerTarget();
                return;
            }

            currentTarget = pomp.workerTarget;
            return;
        }

        if (objectsHolder.HandsIsFull() && _workerState ==
WorkerState.CollectObject)
        {
            _workerState = WorkerState.FillVitrine;
            FindWorkerTarget();
            return;
        }

        if (objectsHolder.ObjectsCountInHand() <
_data.InventoryCapacity && _workerState ==
WorkerState.CollectObject)
```

```
        {
            currentTarget = pomp.FindGrowdFlower();
            if (currentTarget == null)
            {
                _workerState = WorkerState.Watering;
                currentTarget = pomp.workerTarget;
            }
        }

        if (_workerState == WorkerState.Waiting)
        {
            currentTarget = waitingPosition;
        }
    }

    private void FindFillerTarget()
    {
        if (objectsHolder.ObjectsCountInHand() > 0 &&
_workerState == WorkerState.FillMachine)
        {
            if (resourceToTarget != null)
            {
                currentTarget =
Magazines.Instance.GetWorkerTargetByResource(objectsHolder.Get
LastObjectSettings());
            }
            else
            {
                if (_exportMachine == null)
                {
                    _exportMachine =
Magazines.Instance.GetExportMachine();
                }

                currentTarget = _exportMachine.workerTarget;
            }

            return;
        }

        if (objectsHolder.HandsIsEmpty() && _workerState !=
WorkerState.CollectObject)
        {
            _workerState = WorkerState.CollectObject;

            if (resourceToTarget != null)
            {
                if (_objectCreatorMachine == null)
                {
                    _objectCreatorMachine =
Magazines.Instance.GetObjectByResource(resourceToTarget);
                }
```

```
                currentTarget =
_objectCreatorMachine.workerExitTarget;
            }
            else
            {
                if (_honeyZone == null)
                {
                    _honeyZone =
Magazines.Instance.GetHoneyZone();
                }

                currentTarget = _honeyZone.workerTarget;
            }

            return;
        }

        if (objectsHolder.HandsIsFull() && _workerState ==
WorkerState.CollectObject)
        {
            _workerState = WorkerState.FillMachine;
            FindWorkerTarget();
            return;
        }

        if (workerType == WorkerType.Filler &&
objectsHolder.ObjectsCountInHand() > 0 &&
            _workerState == WorkerState.CollectObject)
        {
            if (resourceToTarget != null)
            {
                if (_objectCreatorMachine == null)
                {
                    _objectCreatorMachine =
Magazines.Instance.GetObjectByResource(resourceToTarget);
                }

                currentTarget =
_objectCreatorMachine.workerExitTarget;
            }
            else
            {
                if (_honeyZone == null)
                {
                    _honeyZone =
Magazines.Instance.GetHoneyZone();
                }

                currentTarget = _honeyZone.workerTarget;
            }

            return;
        }
```

```
    }

    public void OnWaterringComplete()
    {
        _workerState = WorkerState.CollectObject;
        FindWorkerTarget();
    }

    public void TakeResource(CollectableObject resource,
Vector3 scale)
    {
        var result = objectsHolder.AddToList(resource, scale);
        if (result)
        {
            workerAnimator.SetBool(Holder, true);
        }

        FindWorkerTarget();
    }

    public bool CheckBeforeTakeResource(ObjectSettings
settings)
    {
        if (resourceToTarget != null)
        {
            return resourceToTarget == settings;
        }

        return true;
    }

    public void FindNextTarget()
    {
        if (workerType == WorkerType.Placer || workerType ==
WorkerType.CottonPlacer || workerType ==
WorkerType.CottonFiller || workerType == WorkerType.Filler)
        {
            FindWorkerTarget();
        }
    }

    public void FindNextTargetFiller()
    {
        _workerState = WorkerState.CollectObject;
        FindWorkerTarget();
    }

    public int ObjectsCountInHand()
    {
        return objectsHolder.ObjectsCountInHand();
    }
```

```
    public CollectableObject GiveResource(ObjectSettings
resource)
    {
        var result = objectsHolder.RemoveFromList(resource);
        if (objectsHolder.HandsIsEmpty())
        {
            workerAnimator.SetBool(Holder, false);
            FindWorkerTarget();
        }

        return result;
    }

    public void VitrineIsFull()
    {
        if (workerType == WorkerType.CottonFiller)
        {
            return;
        }

        _workerState = objectsHolder.HandsIsFull() ?
WorkerState.Waiting : WorkerState.CollectObject;
        FindWorkerTarget();
    }

    public void VitrineIsEmpty()
    {
        if (_workerState == WorkerState.Waiting)
        {
            _workerState = WorkerState.FillVitrine;
            FindWorkerTarget();
        }
    }

    public bool CanTake()
    {
        return !objectsHolder.HandsIsFull();
    }

    private void SetTargetDestination(Vector3
destinationPosition)
    {
        _aiPath.SetPath(null);
        _aiPath.canMove = true;
        _aiPath.maxSpeed = _data.MovementSpeed;
        workerAnimator.SetBool(Movement, true);
        _aiPath.destination = destinationPosition;
        StartCoroutine(CheckReachDestination());
    }

    private void StopMovement()
    {
        _aiPath.SetPath(null);
```

```
        _aiPath.canMove = false;
        workerAnimator.SetBool(Movement, false);
    }

    private IEnumerator CheckReachDestination()
    {
        while (true)
        {
            if (_aiPath.reachedEndOfPath &&
!_aiPath.pathPending)
            {
                break;
            }

            yield return null;
        }

        StopMovement();
        RunEventOnReachDestination();

        yield return null;
    }

    private void RunEventOnReachDestination()
    {
        switch (workerType)
        {
            case WorkerType.Placer:
                break;
            case WorkerType.CottonPlacer:
                break;
            case WorkerType.Casier:
                transform.DORotate(onTargetRotation, 0.2f);
                break;
            case WorkerType.Filler:
                transform.DORotate(onTargetRotation, 0.2f);
                break;
            case WorkerType.CottonFiller:
                break;
            case WorkerType.Wrapper:
                transform.DORotate(onTargetRotation, 0.2f);
                break;
            default:
                throw new ArgumentOutOfRangeException();
        }
    }
}
```

ДОДАТОК В

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	79 с.
	Інші документи	
2.	Презентаційні матеріали	15 плакатів

.	.	Прізвище та ініціали.	Підп.	Дата	
Розробив		Стрельченя Д.Ю.			Використання штучного інтелекту у розробці мобільних додатків
Перевірив		Узлов Д.Ю.			
Н.контр.		Малєєва І.А.			
Затв.		Філатов В.О.			

Шифр групи	Код напр./спец.
СШІм-20-3	122
ХНУРЕ кафедра ШІ	