11/22/2024

XBCGD7312

Scripts Document

Pasqaulo Shields, Sybrandt van Niekerk, Malcom Joe Thonger, Nasser el Quoar, Maxime Declerck, Eugene van den Berg

Worker AI Test

This is a test script we used to test our employee prefabs are moving and working properly inside a sperate test scene.

Functionality.

* **Employee Instantiation:** The SpawnEmployee() method is used to instantiate (create) a new employee NPC in the scene, represented by the employee prefab. Once the employee is instantiated, the script checks if the EmployeeMovement component exists and starts its movement by calling StartMoving().
* **Employee Firing:** The FireEmployee() method is used to make the current employee follow a different path (the "fired path"). It retrieves the EmployeeMovement component and calls the FireEmployee() method to initiate the firing process.

Worker Ai

The WorkerAI script is responsible for controlling employees in the game. It manages their movement along assigned paths and idle behaviors, including their animation states. This is an easier and more direct script used for when spawning the employees in the scene and by default have them sit in the reception area.

Functionality:

* **Path Movement:** The employee can follow a path defined by waypoints (points in the scene the worker moves between). The path is assigned from a list of possible paths (pathObjects). The worker moves along the path using a NavMeshAgent, adjusting its destination for each waypoint.
* **Spawning and Firing:** Employees can be spawned at specific spawn points (spawnPoints) and assigned a path to follow. Employees can be fired (removed from the game) by sending them to an exit point (exitPoints), where they leave and get destroyed.
* **Animation Control:** The employee has an Animator component to control animations based on movement states. The employee’s animations change based on whether it is walking, idle, or sitting. The UpdateAnimator() method manages these transitions.
* **Waypoint Handling:** As the employee moves between waypoints, it either continues to the next waypoint or waits for a specified time (waitTime) at each waypoint, toggling idle animation during the wait. Once the employee reaches the last point, it enters a sitting state at the end of the path.
* **Gizmos:** The OnDrawGizmos() method visually draws the path in the Unity Editor for easier debugging by connecting the waypoints with a red line.

Random Idle and Talk script

The RandomIdleandTalk script controls the switch between two animations, Idle and Talking. It spawns a random pattern, based on a timer. This is for the employees that players won’t really interact with and just populate the scene.

Functionality:

* **Key Features: Animator Control:** The script toggles between the IsIdle and IsTalking animation states by adjusting the parameters of the animator. Initially, the employee is set to Idle (IsIdle = true) and not Talking (IsTalking = false).
* **Timer Logic:** A timer (timer) is used to randomly switch between idle and talking animations. The timer is set to a random value between the minTime and maxTime range. When the timer reaches zero, it toggles between idle and talking states and then resets the timer with a new random value.
* **Random Timer:** SetRandomTimer() generates a random timer value between minTime and maxTime, ensuring the time before the next animation switch is unpredictable.
* **Animation Transitions:** After the timer reaches zero, the animation state switches between Idle and Talking, with the IsIdle and IsTalking flags being toggled**.**

Employee Sit script

The EmployeeSit script controls the sitting behavior of an employee character. This is to have prefabs of the characters just sitting and easily place them around.

Functionality:

* **Animator Control:** The script begins by ensuring the employee is in a sitting state when it starts. It sets the IsSitting animation to true, and IsIdle and IsWalking to false. This guarantees that the employee starts off sitting and not walking or idle.

Employee Movement script

The script manages the movement and behavior of an employee using the NavMesh system and animations. It’s mainly used when spawning the employes in the scene and moving them around or generally making different interaction.

Functionality:

* **Path Following:** The employee follows one of two paths: a *spawn path* when they are introduced into the scene or a *fired path* when they are removed.
* **Animation Control:** Animations are updated dynamically based on the employee's state (moving, sitting, or idle).
* **Spawn Behavior:** After completing the spawn path, the employee rotates to a predefined angle and transitions to a sitting animation at their assigned workstation.
* **Chair Assignment:** Employees can be directed to specific department chairs, where they adjust their height and start a sitting animation upon arrival.
* **Firing Behavior:** When an employee is fired, they follow the fired path, and their object is destroyed when reaching the final waypoint.
* **Obstacle Handling:** The script detects obstacles using triggers and temporarily redirects the employee's path to avoid collisions.

Chair script

The Chair script is designed to handle employees sitting in chairs, as part of the office management system. Each chair will have this script attached

Functionality:

* **Sitting Employees:** The Chair class keeps track of a list of GameObjects called sittingEmployee, which represent different employee models that can sit in this chair. The department variable categorizes which department the chair belongs to. The hasEmployee boolean tracks whether an employee is currently sitting in the chair.
* **Set Employee:** The SetEmployee(int position) method enables the employee model in the sittingEmployee list at the specified position index. It also sets hasEmployee to true to indicate that there is an employee in the chair.
* **Reset Chair:** The ResetChair() method loops through the list of sittingEmployee models, deactivating each one. It also sets hasEmployee to false to indicate that no employee is sitting in the chair.

Chair Management script

The ChairManager script is designed to manage the assignment of employees to chairs based on their position or department.

Functionality:

* **Chair List**: The script has a List<GameObject> named chairs, which holds the chair objects in the game.
* **Assigning Chairs**: The AssignChair method assigns a chair to an employee (My\_CV object) based on their department (e\_position). If the employee has a position other than Unassigned, it iterates through the list of chairs and tries to find an unoccupied chair that matches the employee’s department (currentChair.department == employee.e\_position). Once an available chair is found, it deactivates the employee's GameObject, sets the employee’s current chair, and activates the chair's employee model based on the employee's sittingModelPos. If no position is assigned (e\_position == Unassigned), the employee is reactivated, and no chair is assigned.

Employee Manager script:

The Employee Manager script is designed to manage the process of hiring, firing, and tracking employees. It handles assigning employees to chairs, managing employee stats, and updating the office's happiness and efficiency levels.

Functionality:

* **Employee Management:** Hiring Employees: The **HireEmployee()** method handles hiring employees from the unassigned list, assigning them to a random available chair, and updating the office's happiness and efficiency. Firing Employees: The **FireEmployee()** method allows for removing an employee from the assigned list, freeing up their chair, and recalculating office stats (happiness and efficiency). Declining Employees: The **DeclineEmployee()** method removes an employee from the unassigned list without hiring them.
* **Chair Assignment**: Employees are assigned to randomly available chairs using the **GetRandomAvailableChairIndex()** method, which checks for unoccupied chairs and assigns them.
* **Employee Stats:** The script tracks the total happiness and efficiency of the office by aggregating the stats from all assigned employees. These stats are then passed to the Game Manager to update the global office stats.
* **UI and Notifications**: The script uses CVpage to display the CV page when unassigned employees are available. A notification is triggered whenever there are unassigned employees, and a sound is played when a notification appears.
* **Chair Management**: Chairs are tracked and managed, ensuring that when an employee is hired or fired, the chair is updated accordingly.

Employee Card Manager.

This script manages employee hiring, firing, and assigning chairs, tracks employee status (happiness and efficiency), and handles notifications for the hiring process. It also checks and updates the status of chairs to ensure employees are assigned properly.

Functionality:

* **Singleton Instance:** The Employee Manager class is a singleton, meaning only one instance exists in the game. This instance is accessible globally.
* **Employee and Chair Management:** Holds employees who are currently assigned to a chair. Holds unassigned employees waiting to be hired, list of chairs in the office and a list that tracks whether each chair is occupied.
* **Hiring and Assigning Employees:** The script hires the first unassigned employee from listUnEmployees, assigns them to a random available chair, and updates office happiness and efficiency stats based on the employee's stats. The HireEmployee() method checks for available chairs and assigns the employee to one. It also updates the lists and recalculates the office stats.
* **Firing and Removing Employees:** The script allows firing employees using FireEmployee(), which removes the employee from the listAssigned, frees up their chair, and recalculates office stats. The FreeChair() method disables the employee's model, unmarks the chair as occupied, and handles chair availability.
* **Employee Stats:** The script tracks the average happiness and efficiency of all assigned employees. These values are updated whenever an employee is hired or fired.
* **Notifications:** Displays a notification when there are unassigned employees in the system. The notification visibility is toggled based on whether there are unassigned employees.
* **Chair Availability:** The script ensures that employees are assigned to chairs only if available, using a random chair selection from the list of chairs (GetRandomAvailableChairIndex()).
* **Interactivity:** The hireButton is disabled when there are too many unassigned employees (4 or more) to prevent over-hiring. The button's interact ability is checked via CheckUnassignedSpace().
* **Dropdown Interaction:** Department Dropdown: Tracks the dropdown selection for departments, though the value isn't used in the current script.

Employee Spawner script

The EmployeeSpawner is responsible for spawning employees at a specified location in the game. It controls the delay between spawns, the number of employees to spawn based on the day of the week and interacts with the Employee Manager to add spawned employees to the list of unassigned workers.

Functionality:

* **Start:** Starts the spawning coroutine when the game starts.
* **Update:** Continuously checks if the game day has just started, and if so, updates the spawn amount based on the current day.
* **Spawn Employee**: A coroutine that handles the spawning of employees. It waits for a random delay, checks if spawning is allowed and spawns an employee at the assigned location. It then waits for the next spawn cycle and randomizes the delay.
* **Spawn Amount**: Adjusts the number of employees to spawn based on the current game day (from Monday to Saturday).

CV Manager:

Set up the cv on the GUI for the player and can reset the values.

CV SO:

CV Scriptable object to able to spawn in specific CVs used for the tutorial

My CV:

This script stores and generates the values used in the CV Manager script.  
It’s stored on the employee npc themselves

Employee Stats:

A script to store lists of Enums about employee status. (Abandoned)

Random CV Generator:

Generates a random cv. (Abandoned)

Employee:

Store Enum data list. (Abandoned)

Game manager:

The Game manager, functions as a script that managers the rotation of the Sun ( directional light) in the game by using the time to change the rotation to a certain degree to mimic the real world sun.

The Game manager also functions as the time keeping system of the game. It tracks and increments the time in the game.

Tutorial Dialogue:

The tutorial dialogue manages the Dialogue for the tutorial as well as the transitions between the different images for the tutorial scene.

Speed Up Time:

The Speed up time scrips only functionality is to speed up the in game time.