Design Decisions:
Storing the selected objects in a queue: I wanted users to only be able to select 2 objects at a time and order matters, so a first come first serve data structure seemed like the best way to handle that.

The user can see with a line renderer which object they would be choosing with a click, and that line's material color is dependent on whether the object is Selectable or some other object. Only selectable (objects with the Selectable script) can be moved or compelled to move toward another selectable object. Other objects can only be renamed. The tables and walls, for example are not selectable, because their size would make the box collider of a combined mesh much too large for the player to easily continue to traverse the room.

The first selected object moves toward the second selected object along a nav mesh, not the other way around. Only the second object's box collider is set to trigger, so when the OnTriggerEnter event is called, Unity only attempts to combine the meshes once.

NavMeshes for selected objects' movement: the navmesh is composed of the ground and the tops of the tables. Every table top has a bi-directional off mesh link from the top of the table to the ground, so it's possible for objects placed on different tables or the ground to meet up.

Issues Encountered:
I had a lot of difficulty with combining meshes in that the position of the new mesh was always offset or the scale was very different from either of the original objects' transforms. To fix this, I set the parent to null for each object whose mesh would be added to the combined mesh. I also had to do some matrix transformations for where the mesh of the second object would be in the transformed object. After the combining was finished I set the position, rotation, and local scale of the combined object to those that the original second mesh had. Then I set the parent of the new combined object to be the original parent of the second object.

There was also some difficulties with defining the NavMesh to include the tables as a valid surface. Eventually I found out the agent radius to table size ratio was too small, so I made the agent radius much smaller, baked again, and it worked much better.

Finally, when I built and opened the project as an executable for the first time, some of the UI managed by the game manager wasn't set up for the main scene (never had this kind of error in the editor, which made it even weirder).

Meeting My Requirements:
I have a clipboard with objectives that must be completed within a certain amount of time, the clipboard updates when the user completes something on it (though there is no notification of this happening unless the user opens the clipboard manually with spacebar).

Objects do move around using nav meshes. Where they move is based on the objects selected and the order of selection. When they connect or the nav mesh agent has reached its destination, the objects' meshes are combined.

Raycasting is used to detect which object the user selected.
I did not use joints as I specified in my proposal, I instead used raycasting and nav mesh agents.

The user is shown their "grade" when they complete all the objectives or the timer runs out.

Instructions:
Hold space to see the clipboard and relevant progress in completing the game. (Let go and the clipboard disappears.)

Left-click on an object to select it for npc motion and mesh combination. You can only select objects that are highlighted as green with the pointer/ line renderer.

Right-click on an object to rename it. (can rename any object the pointer can land on)

    Once done typing the new name, press Enter.
    If you decide not to rename the object, press Escape.

Move around with w, a, s, and d (or the arrow keys).

Complete all the objectives listed on the clipboard within the time limit to win the game.