Using and Constructing Elevators

Introduction

This manual is intended to give guidance for those seeking to use the supplied scripts, prefabs and other assets in order to have working elevators in their scenes. This document aims to outline how to use the assets to achieve this.

Please bear in mind that these scripts are supplied as a ‘bonus’ feature of this asset pack. There are better, fully featured elevator systems out there in the asset store. My elevator system of scripts is simple and only designed to show off the asset artwork at its best.

All that being said, this system does work and could provide the basis for a much more sophisticated system with more work. I expect to refine this over time.

Tutorials

At the time of authoring this document the asset pack is not published. Documenting the elevators etc. is one of my last tasks before submission. Once submitted I will forge ahead with recording some on-line video tutorials outlining how to get the best out of the assets. One of these videos will be dedicated to the elevators. These videos will appear here:

http://madewithmice.com/game-assets/clean-sci-fi-modular-corridors-and-rooms/

The Elevator System

I have provided custom meshes and scripts, assembled into prefabs to get you up and running quickly. Although it is possible to build the elevator system from scratch, it is recommended that you use the prefabs provided and modify them as needed. The prefabs are names thus:

Service_Elevator_3_Floors
Personnel_Elevator_3_floors

Both types, as their name suggests, consist of an elevator shaft giving access to three floors. In each case floor 0 (always counting from zero, not one!) is set at 0 in the Y axis. Please bear in mind that this is not a restriction. Your lowest floor could be an entirely different height in the Y axis, I’m merely stating that the logical height for the prefab is 0.

The ‘service’ elevator is an open elevator platform with rails, allowing the passenger to see the elevator shaft as they travel. The ‘personnel’ elevator is a closed box elevator type in which the passenger does not get to see the elevator shaft.

The Prefabs

Both prefabs should work ‘out of the box’. The scripts supplied on the relevant objects in the prefabs have already been ‘wired’ up and are thus pointing to their correct elements. Things should just work.

However, the prefabs and scripts require certain conditions to operate properly. If the scripts/objects have been pre-’wired’ then those objects have already found their correct partners. If they have not been pre-’wired’ then the scripts will automatically try to find their partner scripts. They will ‘self-wire’, if you will. The success of this depends upon the user creating their own prefab assembly in a specific hierarchy. The hierarchical relationship between elements helps the scripts find their partners, and therefore helps the elevators to function properly.

Elevators consist of the following:

- Elevator Managers (master scripts)
- Elevators (the element that moves up and down)
- Elevator lock objects (invisible objects with mesh colliders, turned on and off to keep players from falling out of the elevator when in motion)
- Elevator doors (a parent object of doors attached to the elevator: inner doors or rails)
- Elevator buttons (buttons inside the elevator to choose floors)
- Call buttons (elevator buttons set at each floor to ‘call’ the elevator)
- Floor doors (elevator doors that open at each floor, not attached to the elevator)

Hierarchy breakdown

Elevators are fairly complex systems so I strongly recommend using the prefabs provided. However, if you need to start from scratch then it is fairly straightforward. In as simple language as I can muster it works like this:

Each elevator requires a ‘master’ script called ElevatorManager.cs to be applied to the top of its hierarchy. So (if you are not using my prefabs) start by making an empty game object, give it a sensible name (e.g Elevator_001), set its transform to the base of your intended elevator shaft (lowest floor) and apply the ElevatorManager.cs script component. The elevator platform (or compartment) is a child of the manager object and needs the Elevator.cs script attached. Child objects of the elevator compartment/platform are the buttons (with elevatorButton.cs attached), one for each floor/deck. These buttons are inside the elevator so the operator can choose a floor. Other children of the elevator are the inner doors, which travek up and down with the elevator. These need to be flagged as inner doors using a tick box in their inspector. These doors need a parent game object which is dragged to the elevator manager to make a connection. Also parented to the elevator is a ‘lock’ object. This is a non-renderable (invisible) object that has its collision switched on and off by the script to keep the player safely inside the elevator whilst moving. Again, this needs dragging to the elevator manager script so it knows to turn the collision on and off. Each floor/deck should have an empty game object to

act as parents of the graphic assets to denote each access point. I've named mine numerically "Deck_A" etc., but you can call them what you wish. What matters is that they are placed at the intended height in the Y axis at which you want the elevator to stop. I've used intervals of 8 metres per floor/deck. Tweak the prefabs if you wish to alter the space between floors. Beware that much less than 4m is likely to cause collision issues. These 'floors' need call button child objects that have the elevatorButton.cs script attached. In the inspector for these scripts remember to specify which floor they call. Numbering begins at the lowest floor: 0, 1, 2 etc.

If in any doubt look at how the working examples are set up in my demo and simply try to recreate that in your own level. Use of my prefabs is highly recommended.

Once again I must say that elevators are fairly complex to set up so I thoroughly recommend using the prefabs provided. How to use them and also how to make your own will be covered in some videos recorded and provided here:

http://madewithmice.com/game-assets/clean-sci-fi-modular-corridors-and-rooms/

I know that this document is very rough at the moment but revisions and clarity should follow soon. Thank you for your patience. Good luck, and thank you.