System Control (10 points)

Submission due by Sunday, March 26 at 11:59pm CT

Purpose

To learn how to design and develop a system control technique with a controller and implement different interaction modes

Overview

In this assignment, you will implement two types of 2D floating menus: an object menu for object interaction and a settings menu for controlling the settings.

Directions

- Open your "CS6334 + *your net id*" project and create a new scene called "assignment03".
- Assignment #3 should be built upon Assignment #2. Make sure that you have all the proper prefabs and game objects placed correctly in a new scene. You can reuse the virtual environment that you created for Assignment #2.
- Your scene should contain at least five unique interactable objects. Each interactable object should be different from each other (e.g., table and lamp are two different objects). If you do not have enough unique interactable objects, please create new objects.
- The 2D menu can be created using Canvas and the Button components in Unity UI. Canvas can act as the menu's body. You can use the buttons to make the interactable options. They can be found under the GameObject -> UI tab in the Unity editor.
- There are two menu systems you need to implement - an object menu and a settings menu:

<u>Object Menu</u>

- In the object menu, you need to implement a 2D menu that follows the **Object-Action-Parameter** complex task sequence (please refer to eLearning->Slides->05 Interaction Techniques to learn about task sequences).
- The object menu should be displayed whenever the reticle pointer points at any interactable object, followed by pressing the "X" button in the controller. Here are some requirements for the menu implementation:
    - By default, the menu should not be shown in the scene.
    - When the reticle pointer points (i.e., hovers) at any interactable object, the corresponding interactable object should be highlighted in white color, indicating that the object is interactable. If the button "X" is pressed while the interactable object is pointed, the menu should be displayed next to the selected object.

- If the reticle pointer points at any non-interactable objects, the corresponding object should NOT be highlighted. The menu should NOT be displayed when the button "X" is pressed.

- When the menu is open, the movement of the *Character* should be disabled. However, the rotation of the *Character* is still active, and the player can still look around and interact with any nearby objects using a reticle pointer. If the player opens the menu of the nearby object while the previous menu is open, then the previous menu should be automatically closed.

- There are three menu items in the object menu and the reticle pointer should be able to select any of the menu items by pointing and pressing the "B" button:

  Grab: When the "B" button is pressed to select the "Grab" menu item, the interactable object should be attached to the reticle pointer. The *Character* should be able to move around while the interactable object remains attached to the reticle pointer. The object menu should be disappeared. When button "A" is pressed, the interactable object should be released from the reticle pointer and dropped on the ground.

  Store: When the "B" button is pressed to select the "Store", the selected object should be disappeared and be stored in the Inventory. The object menu should be disappeared as well. The *Character* should be able to move again. The maximum number of objects that can be stored in the Inventory is three. If the Inventory is full (i.e., three objects are already stored), the message "Inventory is full!" should be displayed, and no more objects should be stored. The message should be displayed for 2 seconds. The selected object should not be disappeared. Please see Inventory in the Settings menu for more details.

  Exit: When the "B" button is pressed to select "Exit", the menu should be closed (disappeared). The *Character* should be able to move again.

<u>Settings Menu</u>

- In the setting menu, you need to implement a 2D menu that follows the **Action-first** task sequence (please refer to eLearning->Slides->05 Interaction Techniques to learn about task sequences).

- The settings menu should be displayed when the button "OK" is pressed on the controller. Here are some requirements for the menu implementation:

  - When the menu is open, no other interactions are allowed. The *Character* should not be able to move or rotate. The reticle pointer should disappear.

  - If the object menu is open when the button "OK" is pressed, the object menu should automatically close.

  - The "up" and "down" of buttons of the controller should be used to navigate the menu items. By default, the first menu item is highlighted in yellow color. When the "up" or "down" button is pressed, the menu item that is above or below will be highlighted, respectively. When the "B" is pressed, the menu item will be selected.

- There are four menu items in the settings menu:

  Resume: Pressing this menu item will make the system menu disappear and resume the application. The *Character* should be able to move and rotate, and the reticle pointer should be visible again.

  Inventory: Pressing this menu item will make the system menu disappear and the inventory panel should be displayed with all stored objects. Each object in the inventory should be displayed as its respective thumbnail image (2D or 3D). Each thumbnail image in the inventory panel should be navigated using "left" and "right" buttons highlighted in yellow color. When the "B" button is pressed to select the thumbnail image, the corresponding object should be attached to the reticle pointer (i.e., Grab) to be placed on the ground by pressing "A" button. The corresponding object should be removed from the inventory panel. The player can continue to store the new item in the inventory of up to three objects.

  Speed: Pressing this menu item will toggle between three-speed modes. The default mode is "Speed: High". The high mode will set the movement speed of the *Character* to 20 m/s. The medium mode will set the movement speed to 10 m/s. The low mode will set the movement speed to 5 m/s. The corresponding speed should appear in the menu item (i.e., "Speed: High" if high mode is toggled, "Speed: Medium" if medium mode is toggled, and "Speed: Low" if low mode is toggled). Once the desired mode is toggled, the player can select Resume to go back to the scene.

  Quit: When the "B" button is pressed to select "Quit", the application should close.

- Click on File -> Build Settings -> Build to generate APK file. Save it as "*your net id*_assignment03".apk.

- After building it, save the scene as assignment03.unity.

- Here is the demo video that shows how the final implementation would look like:
  - https://youtu.be/Glkuh5dhgTs

Submission

1. Clean up your Unity project by removing any unnecessary assets from the "Assets" folder and deleting the project's automatically generated "obj", "Library", and "Temp" folders. Your submission must be **500 MB** or less.

2. Record a video showing all the required functionalities from your phone's perspective. You can either use the built-in screen recording feature or download apps like AZ Screen Recorder, OneShot, Unlimited Screen Recorder, etc. The video should not be more than 50 MB and it should not be more than 30 seconds long. Minimum resolution for the video should be 480p (640 x 480).

3. Create a "Source" document (.pdf) that provides a unique URL for where you obtained each virtual object within your project and explains how each of your script works. If the virtual objects are created by you, please indicate that you created them by yourself.

4. Create a zip file (.zip) that contains your entire "CS6334 +*your net id*" Unity project folder, your "Source" document and your demonstration video. Do NOT use any compression file types (e.g., .rar, .7z, .tar) other than .zip. Such submissions will NOT be graded, which will result in 0 points.

5. Submit the zip file on eLearning under Assignments > Assignment #3. There will be unlimited attempts to submit your work, and the last submission will be graded.

Scoring

This assignment will be scored as indicated below. The maximum possible score is 10 points.

- ☐ Grab: 3 points
- ☐ Store: 2 points
- ☐ Exit: 0.5 point
- ☐ Resume: 0.5 point
- ☐ Inventory: 3 points
- ☐ Speed: 0.5 point
- ☐ Quit: 0.5 point

Deductions

Deductions will be applied as indicated below. The minimum possible score is 0 points.

- ☐ The 2D menus behave incorrectly. **1 point per each issue**
- ☐ The button behaves incorrectly. **1 point per each issue**
- ☐ The mode doesn't follow the direction. **1 point per mode**
- ☐ You did not follow the direction correctly. **1 point per instruction step**
- ☐ Your video does not show the complete functioning of the assignment. **0.5 point**
- ☐ You did not submit the video. **1 point**
- ☐ Your submission is late. **2 points per day late**
- ☐ Your submission is not a .zip file. **10 points**
- ☐ Your submission is larger than 500 MB. **1 point per 50 MB over**
- ☐ Your Unity project does not properly work during initial grading. **5 points**
- ☐ Your supplementary files are not of the specified formats or do not contain the specified

information. **1 point per file**

- ☐ You did not follow the specified naming conventions. **0.5 point per file or folder**

Regrade Policy

For programming assignments, you have a window of 7 days (from when we return your assignment) to ask for a regrade. We will not consider any regrade requests outside this window. Regrade requests should be emailed to TA; regrade requests will not be considered unless they contain a clear explanation on why a regrade should be issued. The TA will respond to your regrade request within 72 hours of receiving it.

Academic Integrity

This is an individual assignment. Each student is expected to complete his/her own work. If found guilty of academic dishonesty, you will receive 0 points on this assignment. Below is a list of things that are considered as academic dishonesty or not:

Considered Academic Dishonesty:

- If you download and copy an already developed scene from someone else or from the Internet, it will be considered academic dishonesty. Copying the scene and making changes in it is still considered academic dishonesty.
- Sharing your actual program code with other students is considered academic dishonesty. You must not share the actual program code with other students. You should not ask anyone to give you a copy of their code or, conversely, give your code to another student who asks you for it; nor should you post your solutions on the web, in public repositories, or any other publicly accessible place.
- You must not look at solution sets or program code from other years. Looking at solution sets or program codes from other years is a dangerous practice. Most assignments change in a variety of ways from year to year as we seek to make them better.
- Copying scripts directly from other students or the Internet is considered academic dishonesty. Copying the script and making slight changes (e.g., variable names) still considered academic dishonesty.

Not Considered Academic Dishonesty:

- Materials, prefabs, and 2D/3D objects downloaded from Unity Asset store or other sources are allowed to use if they are not from other students (including current and previous semesters). However, the source of the assets should be clearly described in a Source document.
- You can refer to scripting solutions on the Internet, try and understand it and then write your own scripts. However, the source of the scripting solutions should be clearly described in a Source document.

Every submission will be checked for plagiarism. If found guilty, you will receive 0 points for this assignment without any exceptions, and your case will be reported to the department and/or university for further action.

These descriptions and timelines are subject to change at the discretion of the professor.