Game Link: https://github.com/niranjan-chhawdi/CMPT-461-Assignment-5.git

What Went Right?

From a systems perspective, the modular interaction architecture worked well. The separation between PlayerInteractor, AppleInteractable, PlayerInventory, and GiraffeFeeder allowed responsibilities to remain clean and independent. The trigger-based interaction system using OnTriggerEnter and OnTriggerExit successfully managed contextual interactions without relying on expensive raycasting every frame.

The inventory system updated correctly using encapsulated methods (AddApple, ConsumeApples) instead of directly modifying variables. UI updates were handled centrally in UpdateUI(), which ensured consistency between gameplay state and on-screen information. Scene reloading through SceneManager.LoadScene(SceneManager.GetActiveScene().name) created a reliable restart loop without requiring a separate menu scene.

Audio feedback was also integrated correctly through AudioSource and controlled playback timing using coroutines, which prevented indefinite looping and allowed controlled playback duration.

Common Problems Fixed

1. **FBX Material Locking**

   After importing the fence model from Blender, Unity embedded the material inside the FBX file, preventing direct editing. This required extracting materials via the Import Settings panel to create editable material assets. Additionally, textures had to be manually assigned to the URP/Lit shader's Base Map slot.

2. **Self-Intersecting Mesh Warning**

   Unity reported "self-intersecting polygon" errors during FBX import. This was traced

back to overlapping geometry and duplicate vertices in Blender. The issue was resolved by using “Merge by Distance,” recalculating normals, and cleaning non-manifold geometry before re-exporting.

Learning Reflections

This project significantly improved my understanding of the Blender-to-Unity pipeline, especially regarding material handling and shader configuration under URP. I learned that exporting assets is not just about geometry; textures and materials must be explicitly managed and reassigned inside Unity.

On the programming side, I gained a deeper understanding of state management in gameplay logic. Implementing boolean state guards like alreadyFed prevented unintended repeated execution. Using coroutines for timed events reinforced how asynchronous behavior works in Unity without blocking the main thread.

I also learned the importance of decoupling systems. Separating inventory logic, interaction detection, animation triggering, and feeding logic reduced cross-dependency issues and made debugging significantly easier.

Overall, this project strengthened my ability to debug engine-level warnings, manage UI scaling properly, and design clean interaction systems that are scalable for larger gameplay mechanics.

Name: Niranjan Chhawdi
Student ID: 650404