Task 4: Character Creators

Electronic Arts Software Engineering Virtual Experience Program

Describe how you would design your own modular character creator.

User Interface

The player would interact with the character creator through a dynamic interface that simplifies the user experience. This would be achieved by reacting to selections and minimizing the displayed options. For example, once the player has chosen a hair color, they would be presented only with the hairstyle options in that color. As such, the selection order mandated by the creator could be loosely designed to optimize ongoing consolidation of options. Additionally, the user would be able to view the character at its current stage of completion throughout the customization process.

Backend Design

From a backend, organizational perspective, a modular character creator would benefit from implementing a sophisticated system of object oriented design patterns. In addition to creating a fully functioning software, the goals of such design would be to optimize code maintenance and extensibility by following the SOLID principles. Further, for the best user and developer experience, the program would use high-level abstraction with a closely related user interface to structure the fundamental aspects of the program that are unlikely to change.

The class structure would extract multiple levels of components that influence the character design. The goal would be to reduce classes to single responsibilities that can smoothly integrate and/or combine with other classes using class composition. The overarching component breakdown would roughly look like the following:

- Component layers: Skeletal Mesh, Biological Components, Cosmetic Components
- Component attributes: Size/Shape, Color, Style

To achieve this structure and behavior, the modular character creator would integrate the following design patterns:

- Behavioral Patterns: Strategy, Mediator
- Structural Patterns: Composite
- Creational Patterns: Abstract Factory

A *mediator* pattern would be used to centralize communication between different *strategy* contexts. The concrete *mediator* would receive the character customization requests from the client and pass it to the appropriate *strategy* context. The concrete *strategies* would implement the algorithm specific to the associated components and their specifications. This *mediator-strategy* relationship would allow for the nested level of communication required to employ the multiple layers of component categorization and the varying algorithmic requirements between them.

The *composite* pattern would be used to organize the entirety of the objects inherited from concrete classes that ultimately composite the entire character. Additionally, this structure would display the hierarchy of component layers to developers and enable recursive changes resulting from changes closer to the root of the *composite*. The *abstract factory* method could be used in creation of the composition to permit future extensibility.