Unity C# Shader Management System

Unity Shader System - What I Need

Hey! Looking for a Unity dev who can build me a shader management system in the next 2-3 days that can apply realtime material changes across multiple objects while preserving their base colors and textures.

It needs to work with materials based on different shaders.

The Idea

I've got two shader packs from the Asset Store:

- MK Toon - https://assetstore.unity.com/packages/vfx/shaders/mk-toon-stylized-shader-178415
- **Flat Kit: Toon Shading and Water -** https://assetstore.unity.com/packages/vfx/shaders/flat-kit-toon-shading-and-water-143368

I want to build a system where I can easily switch between different art styles for multiple objects at once, while keeping their original colors and textures intact.

Note: They have different shader properties.

You can create a material-swapping system rather than trying to map properties between incompatible shaders. The key is to maintain the original colors and textures when swapping.

What I'm Looking For

1. **A Material Manager**
    - Automatically find all child objects with materials
    - Figure out which objects have unique materials vs. shared ones
      (By 'unique materials,' I mean materials that are only applied to specific objects and not shared across multiple objects. For example, if I have three trees where two use a green material and one uses a red material, then both green and red are considered unique materials that need tracking.)
    - Keep track of objects with special textures
      (Some 3D models use white/neutral materials with UV-mapped textures that provide the actual color information. It's critical that the system identifies these and never modifies their textures or base colors, as this would destroy their appearance)
    - Update material properties across all objects at once
    - NEVER change the base color or albedo texture
    - Make all other properties easy to tweak in real-time

2. **A Simple Editor Interface**
    - Either a custom inspector on a manager object or a dedicated window (or both!)
    - See changes in real-time right in the editor
    - Pick which objects or groups to modify
    - Clearly show which properties will change and which won't

3. **Easy Preset System with ScriptableObjects**
    - Save style presets as ScriptableObjects
    - Include presets for styles like: Realistic, Cartoon, Toon, Comic, Anime, etc.
    - Quick save/load of settings
    - Make 10 presets for me based on the demo scenes of the assets

4. **HIGHLY detailed Documentation for Each Property**
    - Create a custom attribute that explains shader properties clearly
    - Include descriptions and visual impact at different values
    - Something like this:

```
// ShaderProperty: Explains what this does visually // [ShaderProperty("Descri
ption", "Performance Impact", "Category")] // ValueDescription(value, "What it
looks like at this value") [ShaderProperty("Controls how metallic the surface
looks", "Low", "Surface")] [ValueDescription(0.0f, "Looks like clay or rough s
tone - completely non-metallic")] [ValueDescription(0.2f, "Has the appearance
of wood or leather - slightly shiny")] [ValueDescription(0.5f, "Resembles plas
tic or ceramic - moderately reflective")] [ValueDescription(0.8f, "Appears lik
e wet surfaces or polished metal - very shiny")] [ValueDescription(1.0f, "Look
s like chrome or polished steel - mirror-like")] public float Metallic = 0.0f;
```

For shader property documentation, I'm flexible on implementation. If custom attributes are too time-consuming, well-formatted comments directly in the code are acceptable. The priority is detailed descriptions of visual effects at different values, however they're organized.

Technical Stuff

- **Finding and Organizing Materials**
  - Scan through object hierarchies to find all MeshRenderers
  - Group similar materials together
  - Keep track of which textures are unique
  - Application of new material + customization should also apply to array of Materials (if any)
  - Show updates immediately in the scene view
  - Include a method that works efficiently at runtime

Project Details

- We're using Unity 6 with URP
- This is for an iOS mobile app
- Need clean, well-commented code that performs well

What I'll Provide

- MK Toon shader package (latest version from Asset Store)
- FlatKit shader package (latest version from Asset Store)
- Odin Inspector and Serializer (for building the custom editor)

- A small test Unity package containing:
  - Sample 3D (high and low poly) models with different complexity textures + no textures

Timeline and Budget

I'm looking to get this done quickly - in about 2-3 days if possible.

In the future, I'd be happy to pay more to integrate additional Asset Store shaders into the system. The framework you build now should make that pretty straightforward to add later. (i plan to add more shaders to my collection)

▶ If all goes well, i'll ask you to do it for more shaders in the future!