Prefabs

- Struktur
- Workflow
- Hierarchische Benennung

Struktur

Im Grunde genommen habe ich UI-Elemente in drei verschiedene Klassen aufgeteilt:

- Panels,
- Elemente,
- Views.

Beginnen wir mit View:

- Alles was mindestens eine Grafik (auch Text) hat ist grundsätzlich erstmal eine View,
- View ist die kleinste Einheit in der Hierarchie,
- Beispiel: Ein einzelner Button oder ein Icon.

Eine Gruppe von Views ist ein Element:

- Elemente haben idR. eine View und ggf. weitere Grafiken unter sich,
- Beispiel: Ein Button mit zugehörigem Erklärungstext, eine Gruppe von Buttons.

Ganz oben in der Hierarchie steht das Panel:

- Ein Panel ist die größte Einheit und wird entsprechend nicht geschachtelt,
- Es kann sowohl Elemente, als auch Views und einzelne Grafiken unter sich haben,
- Beispiel: Ein Optionsmenü oder Inventar etc.

Diese Aufteilung soll Ordnung in die ganzen Prefabs bringen, indem sie:

1. Bei der Benennung hilft,
2. Hierarchien und damit auch,
3. Zuständigkeiten vorgibt.

Workflow

Abbildung 1: Vier Main-Menu Buttons

Bei der Arbeit mit Prefabs orientiere ich mich an dem Workflow in Figma. So kann man stumpf für jede Master-Komponente aus Figma ein eigenes Prefab erstellen. Solche Prefabs nenne ich "Templates". Templates sind idR. nicht dafür gedacht wirklich selbst verwendet zu werden, sondern um als Ausganspunkt für Prefab-Varianten zu dienen. Ein Beispiel für den Aufbau eines Templates für die Buttons aus Abbildung 1 kommt im Folgenden:

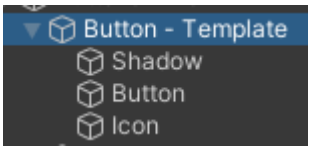

Abbildung 2: Button Template

Grundsätzlich verfügt jeder der vier Buttons über ein Icon, eine Basis und einen Schatten. Dinge, wie die Grafiken der Icons oder Farben des Buttons kann man dann auf einer Prefab-Variante ausdefinieren oder innerhalb eines Elements bzw. Panels, wenn diese Ausprägung nur einmalig dort vorkommt. Auf diese Weise erhält man die selbe Funktionalität, wie die Master-Komponenten in Figma: Größere Änderungen am Design des Buttons sind schnell einzupflegen, weil sie sich auf alle Varianten auswirken, die keine im Konflikt stehenden Änderungen aufweisen.
Wenn man jetzt beispielsweise all unseren Standardbuttons die Notifications des gelben bzw. orangenen Buttons aus Abbildung 1 hinzufügen wollen würde, muss man lediglich das Template anpassen. Die Notification ist wiederum selbst ein Prefab, was im Standardbutton geschachtelt ist, da wir sie womöglich auch außerhalb des Buttons verwenden wollen:

Abbildung 3: Nested Notification Prefab im Button Prefab

Aber wahrscheinlich wollen wir eher eine neue Variante des Buttons einführen, der über die zusätzliche Notification verfügt. Unter Umständen könnte es sogar ein neues Template werden (Wie der Name in Abbildung 3 andeutet). Hier ist die Frage viel mehr, wie kleinteilig und komplex man die Vererbung der verschiedenen Prefabs haben will. Dabei muss jeder für sich selbst die richtige Mitte zwischen Iterationsgeschwindigkeit und Übersichtlichkeit finden.
Oft lohnt es sich lieber noch ein Prefab oder eine Variante mehr zu erstellen, anstatt zu viel in einem einzelnen vereinigen zu wollen. Versionen für links und rechts sind ein gutes Beispiel dafür: Es scheint zunächst verlockend, die Grafiken und das Layout innerhalb des gleichen Prefabs einzubauen, man spart sich so eine weitere Datei, die es zu verwalten gilt, aber beim Animieren sind rechts und links oft genug zu unterschiedlich - deshalb lieber eine Variante links und eine Variante rechts.

Schlussendlich versuche ich innerhalb meines Projektes möglichst wenige lokale Änderungen zu haben, da diese eingehende globale Änderungen von Templates blockieren. Nach Möglichkeit sollten Änderungen lieber eine eigene Variante oder ein neues Prefab werden. Wie immer gilt, dass von Fall zu Fall zu entscheiden ist. Änderungen innerhalb von Szenen sind besonders nervig für Versionskontrolle, weshalb ich die gesamte UI innerhalb von eigenen Prefabs produziere. Es gibt also z.B. kein Button-Prefab, das nur in einer Szene eine andere Farbe annimmt. Einzig die fertige Hierarchie bzw. das Layering findet am Ende in der Szene statt.

Figmas Styles lassen sich leider nicht so einfach umsetzen, ohne eigenen Code zu schreiben. Einzig bei TextMeshPro liegt schon eine Implementierung seitens Unity vor. Grob gesehen reichen Prefab-Varianten für einige Anwendungsfälle von Styles aus - aber mit einem Klick die Highlight-Farbe im gesamten Projekt zu ändern bleibt leider aus.

Hierarchische Benennung

Für die Benennung von meinen UI-Prefabs benutze ich lose das von Masahiro Sakurai vorgeschlagene Schema.

Im Grunde genommen geht es darum selbst sortierende Namen zu erzeugen, indem man eine Hierarchie innerhalb der Namen einhält. Ein Beispiel:
Nehmen wir an, dass wir den Button zum Schließen des Optionsmenü benennen wollen - ein möglicher Name wäre "Menu Options Button Close" mit Leerzeichen oder einfach Camel-Case "MenuOptionsButtonClose". Der Name entsteht durch eine Verkettung von Kategorien, die den Kontext der Verwendung immer weiter eingrenzen:

1. Menu - verwendet innerhalb eines Menüs
2. Options - genauer gesagt im Optionsmenü
3. Button - es handelt sich um einen Button
4. Close - genauer gesagt um den Close-Button

Für Varianten oder fortlaufende Teile einer Animation kann man noch Buchstaben bzw. Nummerierung anfügen: Close Button a-c oder Attack Forward 01 - 05.

Der direkte Vorteil an dieser Benennung ist, dass sie innerhalb eines Ordners die Sortierung vorgibt. Beispielsweise werden alle Elemente des Optionsmenüs nebeneinander sein:

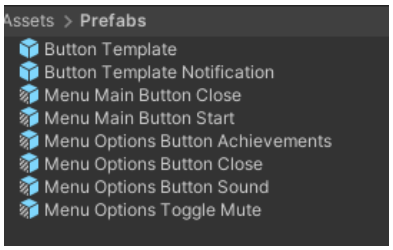

Abbildung 4: Prefabs gruppieren sich anhand der Namen

Bei Anzahl bzw. Umfang von Kategorien kann man nicht pauschal sagen, wie viele Kategorien benötigt werden bzw. wie groß der Umfang sein sollte. Die Entscheidung ist also ein persönliche Präferenz, aber auch vom Projektumfang abhängig.
Aus dem Namen "Menu Options Button Achievements" ist bereits ersichtlich, dass diese Namen sehr lang werden können. Möglichkeiten damit umzugehen wären:

1. Camel Case verwenden um Leerzeichen zu sparen,
2. Kategorien durch Ordner ersetzen,
3. Abkürzungen verwenden.

Meiner Meinung nach würde Camel Case die Lesbarkeit zu sehr einschränken. Sakurai verwendet im Video Abkürzungen. Diese sparen Platz, allerdings haben Abkürzungen immer den Nachteil, dass man sie erklären muss. Sobald also mehrere Leute an dem Projekt arbeiten, müsste man eine Art Glossar für die Abkürzungen aufsetzen. Deswegen ist es meist unkomplizierter die Sachen einfach auszuschreiben. Also würde ich lieber probieren Kategorien nicht zu weit zu fassen oder sie durch Ordner zu ersetzen.

Ein Faktor, den man dabei im Blick behalten sollte, ist die globale Suche: Gibt man "Options" in die Suche ein, erhält man alle zu dem Optionsmenü gehörigen Elemente. Wenn man die Kategorie "Options" in einen Ordner verwandelt hat, findet man jetzt nur diesen. Es kann ebenfalls vorkommen, dass es Kollisionen gibt, wenn es beispielsweise mehrere Close Buttons in den zugehörigen Ordnern (z.B. Options, Shop) gibt - wird hier nach "Close Button" gesucht, erscheinen zwei Assets und man kann nur noch durch Blick auf Pfad oder Inspektor feststellen, welcher der richtige ist.