Layering

- Basics
  - Sorting Group
- Partikel in der UI
- Beispiele

Basics

Unter Layering verstehe ich die Aufteilung von Objekten in Ebenen ("Layers"), insbesondere um Vordergrund und Hintergrund zu unterscheiden. Gerade für Dinge wie Alerts oder andere Arten von Pop-ups ist es für die UI wichtig, eine solche Sortierung über Layering sicherzustellen.

Abbildung 1: Der rote Alert sollte immer über allen anderen Elementen erscheinen.

Wie können wir also sicherstellen, dass unser Alert immer vorne ist, so wie es in Abbildung 1 zu sehen ist? Im Kapitel über das Canvas ging es bereits darum, wie man GameObjects innerhalb eines Canvas sortiert und dann wiederum Canvasses untereinander mittels Canvas-Render-Modes oder Sortieroptionen sortiert. Gegebenenfalls muss die UI sich auch noch gegen Objekte der Welt sortieren - das ist vor allem für Partikeleffekte interessant - weswegen es sinnvoll ist die Hierarchie und das Verhalten von Sortierfunktionen der Renderer zu kennen.

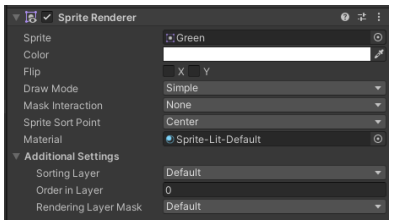

Abbildung 2: Die Komponente Sprite Renderer. Unten sieht man die Sortieroptionen.

Für Layering von transparenten Objekten (z.B. Canvas, Sprites) gilt folgende Hierarchie:

1. **Sorting Layer**
2. **Sorting Wert**
3. **Distanz zur Kamera** (Transparenzachse; standardmäßig entlang Z)

Bedeutet also:

1. **Sind zwei Objekte auf dem gleichen Sorting-Layer?**
   a. Ja - Sorting-Wert wird betrachtet.
   b. Nein - Das höhere Layer wird vorne angezeigt.
2. **Haben die Objekte den gleichen Sorting-Wert?**
   a. Ja - Die Z-Position wird betrachtet.

    b. Nein - Der höhere Sorting-Wert ist vorne.
3. **Der geringere Z-Wert ist vorne.**

Anmerkungen:

- **Diese Hierarchie ist vereinfacht**
  Es gibt noch diverse andere Möglichkeiten Renderer zu sortieren. Die genaue Aufschlüsselung findet sich hier.
- **Es gibt standardmäßig 2 Render Queues "Opaque" und "Transparent", welche auch in dieser Reihenfolge sortiert werden.**
  Zur Erinnerung: Canvas und Sprites sind in der transparent Queue,
- **Die Transparenzachse kann man beliebig einstellen**
  z.B. auf Y an Stelle von Z. Das ist beispielsweise hilfreich für 2D Top-Down-Spiele.

Um die Sortierung von Weltobjekten zu erleichtern gibt es die Sorting Group, um welche es als nächstes geht.

Sorting Group

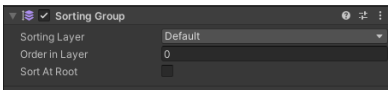

Abbildung 3: Die Komponente Sorting Group

Mit der Sorting Group gruppiert man ein GameObject und seine Childs als Einheit für das Rendern, ähnlich wie es das Canvas tut. Ein Beispiel dafür wäre ein Charakter, der aus mehreren, einzelnen Sprites besteht. Außerhalb der Gruppe gelten jetzt die Sortieroptionen bzw. die Z-Position des Gruppen-Parents für alle Childs. Innerhalb werden Renderer wiederum nach den für sie geltenden Regeln sortiert: Beispielsweise gelten für Sprites Sorting-Layer und Wert, während Z zugunsten der Hierarchie ignoriert wird (erneut, wie beim Canvas). Bei 3D-Renderern wird nach Z sortiert und bei gleichem Z-Wert der Hierarchie nach.

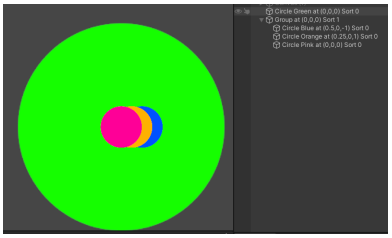

Abbildung 4: Sorting Group sortiert gegen Sprite

In Abbildung 4 sieht man beispielhaft die Anwendung einer Gruppe, um die drei kleinen Kreise zu gruppieren und diese gemeinsam gegen den grünen Kreis zu sortieren. Die Gruppe befindet sich vorne, da sie den höheren Sortierwert (1 gegen 0) hat. Innerhalb der Gruppe haben alle Kreise den gleichen Sortierwert, entsprechend werden die Kreise der Hierarchie nach sortiert. Diese Sortierung ist abweichend von den Z-Werten, da diese ignoriert werden.

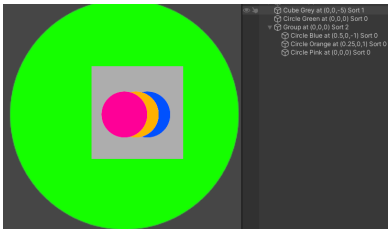

Abbildung 5: Ein grauer Würfel wird einsortiert

Insbesondere bei 3D-Renderern ist wichtig, dass Sortieroptionen nur auf transparente Geometry angewendet werden. Wird der Render-Mode "Opaque" verwendet sortieren 3D-Renderer mit Sorting-Groups trotzdem über die Z-Position gegen andere Sorting-Groups. Wenn der richtige Render-Mode verwendet wird, kann man nun mit der Sprting-Group-Komponente einen Sortierwert zuweisen. Auf diese Weise landet der graue Würfel in Abbildung 5 zwischen den Elementen aus der vorherigen Abbilung 4.

Abschließend sind Sorting groups also für zwei Anwendungen besonders wichtig:

1. Gruppieren von Objekten, die optisch eine Einheit sein sollen – z.B. Charakter aus mehreren Sprites,
2. Anwendung von Sortieroptionen auf (transparente) 3D-Renderer.

Partikel in der UI

Partikeleffekte für die UI sind in Unity von Haus aus nicht sonderlich gut unterstützt. Man kann zwar ein herkömmliches Partikelsystem an Childs eines Canvas hängen, aber das Rendern funktioniert z.B. nicht mit dem Modus Overlay:

Abbildung 5a: Partikelsystem als Child des grünen Play-Buttons

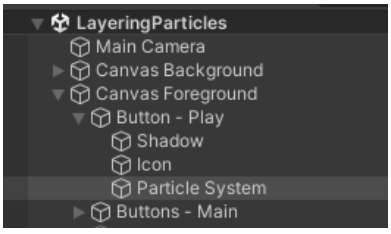

Abbildung 5b: Hierarchie zu 5a

In Abbildung 5a sehen wir, dass die Partikel nicht nur hinter den UI Elementen des Parent-Canvas "Foreground" verschwinden, sie liegen auch hinter dem darunterliegenden Canvas "Background". Wie kommen wir trotzdem an unser Ziel? Am leichtesten wäre es die anderen beiden Render-Modi "Camera" oder "World" zu benutzen. Das bringt jedoch auch deren Eigenheiten mit sich. Im Folgenden möchte ich zwei Ansätze vorstellen, mit denen trotzdem "Overlay" verwendet werden kann. Beide benutzen dafür eine zusätzliche Kamera.

Als erstes kann man die separate Kamera ausschließlich die UI rendern lassen und sie dann als Overlay-Kamera zur MainCamera hinzufügen. Jetzt kann man das Canvas auf "Camera" umstellen, damit es das Partikelsystem korrekt rendert und behält dabei den Overlay-Effekt, sowie Layouting für di Partikel. Bei Verwendung eines einzelnen (Root-)Canvas kann man das Canvas auch auf "Overlay" lassen und das Partikelsystem schlichtweg an der korrekten Stelle in der Welt platzieren.

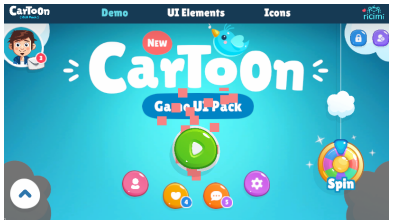

Abbildung 6: Rote Partikel innerhalb eines Raw Image

Anstatt die ganze UI von der neuen Kamera rendern zu lassen, kann man auch nur das Partikelsystem rendern lassen und die Ausgabe der Kamera auf eine Rendertextur geben. Diese Rendertextur kann dann von der UI-Komponente "Raw Image" angezeigt werden. Das hat den Vorteil, dass der Effekt leicht maskiert werden kann, entweder über Masken-Komponenten oder die Größe der Textur. Es kann allerdings einiges an Tüfteln mit den Einstellungen der Rendertextur benötigen, bis man die richtigen Maße und Qualität für den eigenen Anwendungsfall raus hat. Man kann in Abbildung 6 z.B. sehen, dass oben unter dem ersten "o" ein roter Partikel abgeschnitten wird, weil dort die Grenze des Raw Image erreicht ist.

Mehrere Kameras kann man sich allerdings nicht auf jeder Plattform leisten. Ein weiterer Performance-Grund, warum Partikel besser nicht in der UI sein sollten, ist, dass Partikelsysteme meist konstant in Bewegung sind (mehr dazu im Kapitel Performance). Deswegen ist es besser Partikelsysteme in der Welt zu belassen, wenn das Spiel bzw. der Szenenaufbau dieses erlaubt.

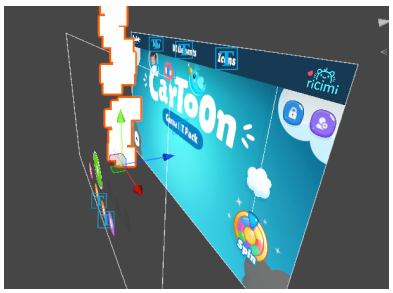

Abbildung 7a: Szenen-Ansicht des Screen Space Camera Setups

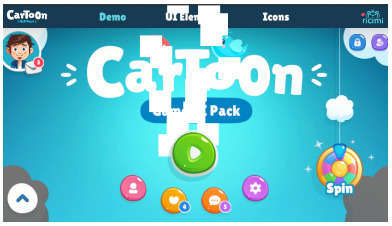

Abbildung 7b: Ansicht im Spiel

Für die Abbildung 7a habe ich die beiden Canvasses auf "Screen Space - Camera" umgestellt und das Partikelsystem dazwischen geschoben. Im Gameview (Abbildung 7b) tauchen die Partikel nun ordnungsgemäß zwischen Vorder- und Hintergrund-Canvas auf. Bei diesem Setup entsteht natürlich die Schwierigkeit, dass man das Partikelsystem korrekt platzieren muss. Das Ganze für statische Elemente per Hand aufzusetzen erscheint noch machbar, aber sobald Dinge dynamisch werden (Bildschirmgrößen bei Mobile, Partikelsysteme dynamisch instanziieren) braucht es wieder ein selbstgeschriebenes System.

Zuletzt noch ein **sehr gutes Package**, was eine kostenlose Lösung für Partikel in der UI bietet: hier.

Beispiele

Wie man seine UI am besten layered ist sehr stark vom Spiel abhängig. Um eine Idee davon zu haben kann man sich beispielsweise folgende Fragen stellen:

- Ist die UI immer vorne?

Die UI könnte ein klassisches HUD sein. In diesem Fall nutzt man ganz einfach den Modus "Screen Space - Overlay" und sortiert verschiedene Canvasse über den Sorting Wert.

- Benutze ich eine andere Achse als Z für die Sortierung von Transparenz?

In diesem Fall würde ich keinen "Screen Space - Camera" Modus verwenden. Diese Art von Canvas sortiert sich immer nach Abstand zur Kamera auf Z - ihr X- und Y-Wert ist immer 0. Entsprechend könnten Weltobjekte nun ungewollt vor der UI erscheinen und es wird unmöglich die Sortierung im Scene View nachzuvollziehen.

- Soll UI dynamisch Spielobjekten folgen?

Canvas mit "World Space" ist dein Freund. Sortiert werden kann das Canvas dann mittels Abstand zur Kamera oder Sortieroptionen.

- Was ist wenn UI hinter Spielobjekten auftauchen soll?

Angenommen wir haben ein 2D-Spiel mit Seitenansicht. Es wäre denkbar, dass z.B. Texte zwischen Spielebene und Hintergrund auftauchen sollen. Für diesen Effekt sollten wir "Screen Space - Camera" oder "World Space" verwenden. Mit beiden Optionen können wir entsprechend unserer Szene die UI entweder über Tiefe oder Sortieroptionen einsortieren. Wenn sich diese UI dem Bildschirm oder der Kamera anpassen soll ist "Screen Space - Camera" zu verwenden.

Wenn UI gegen Welt sortiert wird stellt sich dann die Frage nach der Sortierung:

- Benutze ich Sortieroptionen oder Distanz zur Kamera bzw. Transparenzachse Z?

Meiner Meinung nach schließen sich diese beiden Optionen ein wenig aus und bieten unterschiedliche Vor- bzw. Nachteile. Das liegt vor allem an der zuvor beschriebenen vereinfachen Hierarchie: Sortieroptionen werden immer vor Z angewendet. Letztendlich muss die Wahl hier vor allem anderen mit der Sortierung der Spielszenen funktionieren.

Ich halte folgende Kombinationen für sinnvoll:

- Alles sortiert sich nach Tiefe,
- Alles sortiert sich mit den Sortieroptionen,
- Global werden Sortieroptionen verwendet, aber UI sortiert sich intern nach Z.

Eine Sortierung nach Tiefe bietet vor allem den "What you see is what you get"-Vorteil. Wenn wir im Scene View auf 3D schalten, können wir unser Layering ganz bequem überblicken, so zu sehen in Abbildung 7a. Die verschiedenen Ebenen der Spielszene könnten mit einer Sorting Group versehen werden, damit man diese intern über Sortieroptionen sortieren kann. Nachteil ist natürlich, dass sie nicht funktioniert, wenn Z nicht die Transparenzachse ist - also für Top-Down eher nicht zu gebrauchen. Wenn die Transparenzachse oder die Szenenhierarchie eine Sortierung per Tiefe nicht zulässt, eignen sich die Sortieroptionen am besten, auch wenn sie die Sortierung unübersichtlicher machen.

Ein Mittelweg kann es sein allein die Sortierung der UI über Z zu regeln und für die Welt Sortieroptionen zu nutzen. Dafür würde man alle Canvasses auf ein Vordergrundlayer verschieben, das über dem Weltlayer liegt. So behält man sich den Überblick der Tiefensortierung mittels Sceneview bei.