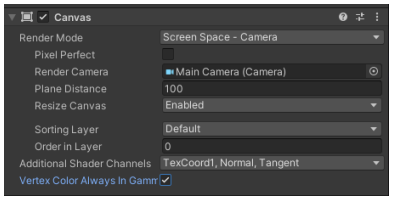
Canvas

- Canvas
  - Canvas Render Modes
    - Screen Space - Overlay
    - Screen Space - Camera
    - World Space
  - Canvas Scaler
  - Nested Canvas

Canvas



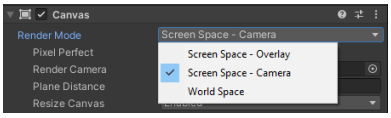
Abbdilung 1: Canvas-Komponente

Jedwede Art von GUI-Element muss innerhalb eines Canvas liegen. Es kann mehrere dieser Canvasses geben. Sie haben eine rechteckige Form und sind ein GameObject mit der Canvas-Komponente darauf. Alle Objekte im Canvas werden in der "Transparent"-Render Queue gerendert.

Innerhalb eines Canvas wird das unterste Objekt im Vordergrund gerendert - die Reihenfolge ist der Hierarchie nach von hinten nach vorne. Das passiert auch unabhängig von der Z-Position. Zusätzlich haben UI-Komponenten, wie etwa Image, auch keine Sorting-Optionen, die man von der Sprite-Komponente gewohnt ist. Entsprechend ist die einzige Möglichkeit UI im Canvas zu sortieren ihr Index in der Hierarchie.

Außerhalb des Canvas verhält es sich wie ein einziger Sprite - egal wie viele Objekte unter dem Canvas liegen und auf welchen Positionen sie sich befinden. Für das Sortieren stehen entsprechende Sortieroptionen analog zum Sprite-Renderer zur Verfügung. Die Render-Reihenfolge von Canvas und Welt-Objekten wird dann noch weiter von den Canvas Render Modes beeinflusst.

Canvas Render Modes



Abbdilung 2: Canvas Render Modes

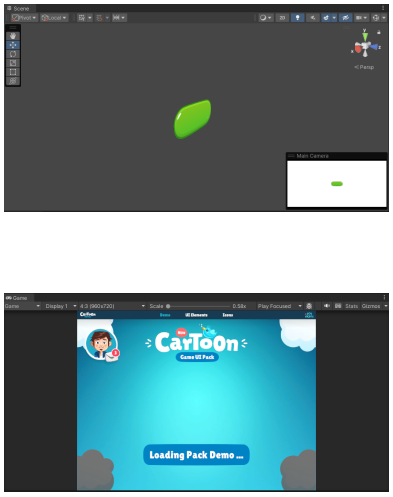
Die drei Render Modes sind:

- Screen Space - Overlay,
- Screen Space - Camera,
- World Space.

Grundsätzlich wird hier unterschieden, ob die UI in Screen oder World Space gerendered wird. Ersteres bedeutet auch, dass sich die Maße des Canvas basierend auf denen des Bildschirms verändern. Ein Screen Space Canvas probiert nämlich immer die gleichen Maße wie der Bildschirm anzunehmen. Das World Space Canvas hingegen hat eine feste Größe, die ihm manuell zugewiesen werden muss.

Screen Space - Overlay

In diesem Modus wird das Canvas nach dem Rendern der Szene über diese gelegt. Sie befindet sich also immer vor allen Objekten, die sich nicht innerhalb weiterer Overlay Canvasses befinden. Weil das Canvas so immer ganz oben gezeichnet wird, entfällt die Option "Sorting Layer" - es bleibt lediglich der Sortierwert zum Sortieren der Overlays untereinander.



Abbdilung 3: Das Canvas wird über den Button drüber gezeichnet

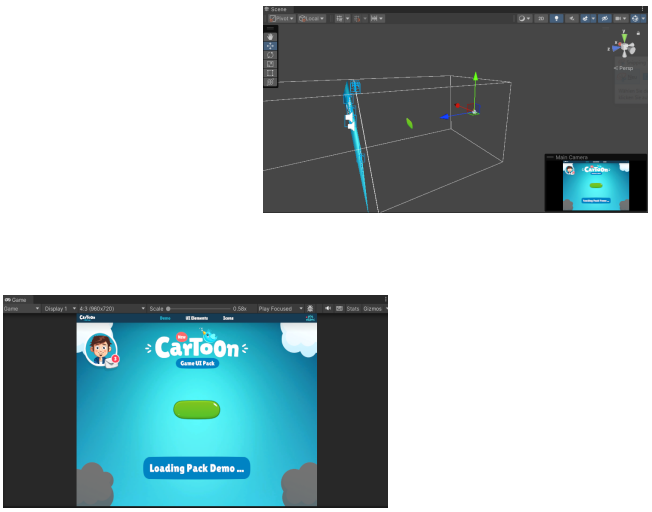
In Abbildung 3 sieht man einen in der Welt platzierten grünen Button. Im Kamera-Preview kann ihn ganz klar sehen - im Game View bleibt er jedoch verdeckt, weil das Overlay-Canvas am Ende über alles gelegt wird.

Diese Art von Canvas eignet sich meiner Meinung nach vor allem für Heads up Displays jedweder Art oder 3D-Spiele, da man sich dort auf der Z-Achse bewegt - was für die beiden folgenden Render-Modes relevant ist.

Screen Space - Camera

Im Modus "Camera" wird die UI in einer festen Distanz vor der zugewiesenen Kamera platziert. Diese Kamera ist dann auch für das Rendern der UI zuständig - entsprechend beeinflussen Kamera-Einstellungen (z.B. perspektivisch vs. orthografisch) das Canvas, ähnlich wie es der Bildschirm tut. Der Abstand zur Kamera und damit letztendlich die Z-Position des Canvas wird über den Parameter "Plane Distance" auf dem Canvas gesteuert (siehe Abbildung 1).

Dadurch entsteht der wichtigste Unterschied zum Overlay: Die Z-Position des Canvas ist nun für die Render-Reihenfolge der Szene relevant. Genauer gesagt können nun Objekte in der Welt vor der UI angezeigt werden, sollten sie sich näher an der Kamera befinden.



Abbdilung 4: Mit Screen Space Camera wird der Button sichtbar

Man kann in Abbildung 4 sehen, dass der grüne Button zwischen Kamera und Canvas liegt. Dementsprechend wird er in diesem Render-Modus vor dem Canvas angezeigt. Des Weiteren ist der Canvas-Inhalt jetzt im Kamera-Preview sichtbar, weil die Main Camera als Camera auf dem Canvas zugewiesen ist. Jedes Screen Space Camera Canvas wird nur von der zugewiesenen Kamera gesehen. Man könnte also kameraspezifische UI bauen.

Wichtig: Die Plane Distance beeinflusst nicht die Größe der UI. Sie ist abhängig von Bildschirm und Kamera. Am einfachsten lässt sich dieses Canvas also über die Plane Distance sortieren, was ebenfalls eine Visualisierung des Layerings im Scene View ermöglicht.

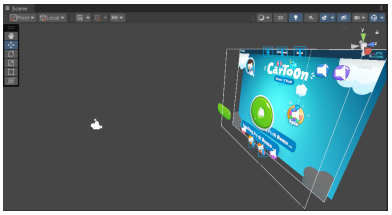


Abbildung 5: Ein zweites Canvas mit Buttons wurde hinzugefügt.

Im Scene View von Abbildung 5 ist sichtbar, dass das zweite Canvas mit den Buttons vor dem bisherigen Canvas liegt, aber immer noch hinter dem grünen Button in der Welt.

Screen Space Camera ist denke ich der beste Render-Mode für Spiele, in denen die Kamera eine feste Distanz zum Geschehen beibehält oder orthografisch ist. Bei Spielen mit Bewegung in die Tiefe besteht immer die Gefahr, dass unsere UI verdeckt wird, weswegen dieser Modus für 3D eher ungeeignet ist.

Wenn man trotzdem seine UI gegen die Welt sortieren möchte, kann man den nächsten Modus nutzen.

World Space

Ein World Space Canvas verhält sich wie jedes andere GameObject auch - wenn es näher an der Kamera ist wird es vorne gezeigt und wenn nicht, dann eben hinter anderen Sachen. Gängige Sortieroptionen bleiben weiterhin erhalten (Wert, Layer). Im Vergleich zu vorangegangenen Modi müssen Breite und Höhe des Canvas nun manuell gewählt werden - sie sind nicht mehr abhängig von Bildschirm oder Kamera. Des Weiteren beeinflusst die Nähe zur Kamera jetzt die Größe der UI-Elemente.

Abbildung 6: Das zweite Canvas wurde auf den Modus "World" gestellt

Abbildung 6 verdeutlicht, dass das Canvas nun wie ein wirkliches Weltobjekt funktioniert: Das zweite Canvas hat eine deutlich kleinere Größe als zuvor und ist damit auch kleiner als das erste Canvas. Durch die Nähe zur Kamera erscheint es trotzdem riesig.

World Space UI kann vor allem hilfreich sein, um sog. diegetische UI in Spielen mit variabler Tiefe zu erstellen. Ein beliebter Anwendungsfall sind Lebensbalken über den Köpfen von Einheiten in Echtzeitstrategie oder Aktion-Rollenspielen.

Unity Manual:
Canvas Scaler

Canvas Scaler

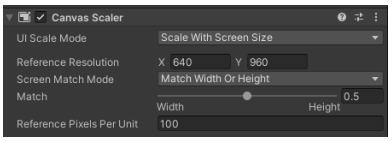


Abbildung 7: Canvas Scaler im Modus "Match Width Or Height", ausbalanciert

Der Canvas-Scaler ist dafür da das Canvas abhängig von der Auflösung des Bildschirms zu skalieren, damit die UI auf verschiedenen Geräten ähnlich bis gleich aussieht. Im Kapitel über Anker wurde bereits angerissen, wie man die Positionierung von UI-Elementen für wechselnde Auflösungen tauglich macht. Wenn wir für das gleiche Setup ins Extreme gehen und sehr viel kleinere oder größere Auflösungen wählen, kann unsere gesamte UI zu groß oder zu klein erscheinen:

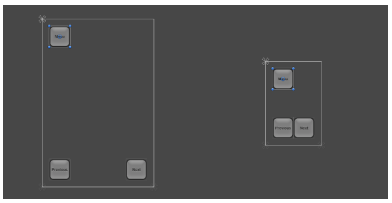


Abbildung 8: Die rechte Auflösung ist um ein Vielfaches zu klein

In Abbildung 8 berühren sich unsere Buttons, da sich ihre Größe nicht verändert hat, obwohl weitaus weniger Platz auf dem Canvas zur Verfügung steht.

Um dieses Problem zu beheben benutzt man den Canvas-Scaler. Dort setzen wir den Modus "Scale With Screen Size" und geben die Design-Auflösung als Referenz an (Abbildung 7). Wenn jetzt die Auflösung einen Wert kleiner oder größer als die Referenz-Auflösung annimmt, wird das Scaling vom Canvas angepasst. Befänden wir uns beispielsweise mit der Referenz 1080p auf einem 4k-Display, würde das Canvas einen Scale von (2,2,2) erhalten, damit alle UI-Elemente mitwachsen.

Es gibt keine Einstellung für den Scaler, die Pauschal am besten ist. Man muss immer auf verschiedenen Auflösungen testen. Beispielsweise könnte es Sinn machen den Wert für "Match" (Abbildung 7) in Richtung Width (0) zu verschieben, wenn man weiß, dass sich die Endgeräte hauptsächlich in der Breite unterscheiden - gleichermaßen würde es für ein Spiel das ausschließlich im Portrait-Modus ist Sinn machen Height (1) zu verwenden, da die Bildschirme sich am meisten in der Höhe unterscheiden werden.

Für das Beispiel Setup produziert die Einstellung Width (0) jedenfalls ein fehlerhaftes Bild:

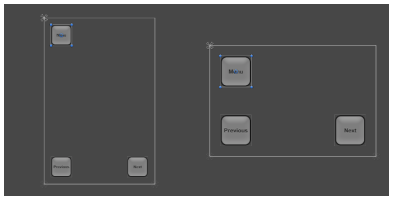


Abbildung 9: Portrait und Landscape mit Skalierung basierend auf der Breite

Da der Landscape-Modus mit 960 x 640 deutlich breiter ist als die Referenz-Auflösung für Portrait, werden die Buttons in Landscape nun größer dargestellt. Wenn die UI also Portrait und Landscape oder auch Tablets vernünftig unterstützen soll, wählt man lieber den Faktor 0,5 (ausbalanciert).

Es gibt noch weitere Screen Match Modes, die ich an dieser Stelle nicht erklären möchte, da dies zu umfangreich wäre und man eh die beste Einstellung für die Zielplattform heraussuchen muss. "Match Width Or Height" ist aber ein guter Start für Mobile.

Beispiel zu Nested Canvas

Nested Canvas

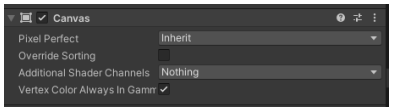


Abbildung 10: Die Canvas-Komponente ändert ihre Ansicht, wenn sie geschachtelt ist

Es besteht die Möglichkeiten Canvasses zu schachteln. Ein geschachteltes Canvas übernimmt die Werte seines Parent-Canvas und bietet entsprechend weniger Einstellungsmöglichkeiten (Abbildung 10). Der Hauptnutzen von geschachtelten Canvasses ist es, dynamisches von statischen Elementen zu trennen. Jedwede Veränderung innerhalb eines Canvas führt unter anderem dazu, dass das gesamte Canvas neu erstellt wird. Mit Schachtelung kann man bewirken, dass nur die dynamischen Teile - also, was sich wirklich verändert hat - neu berechnet werden muss. Die statischen Elemente außerhalb des Sub-Canvas, bleiben unberührt.