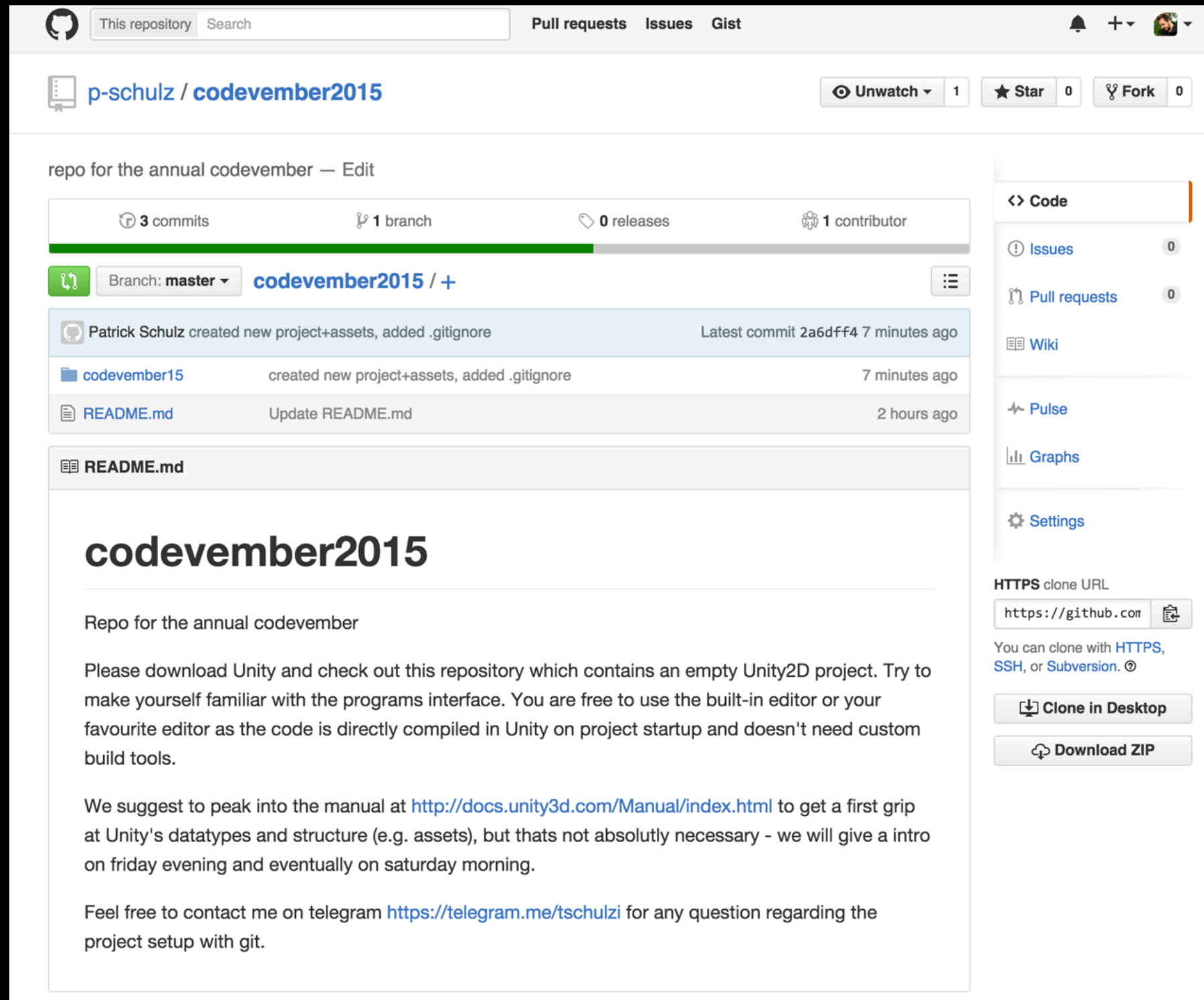




Eine kurze Einführung in Einrichtung, Oberfläche und Funktionen

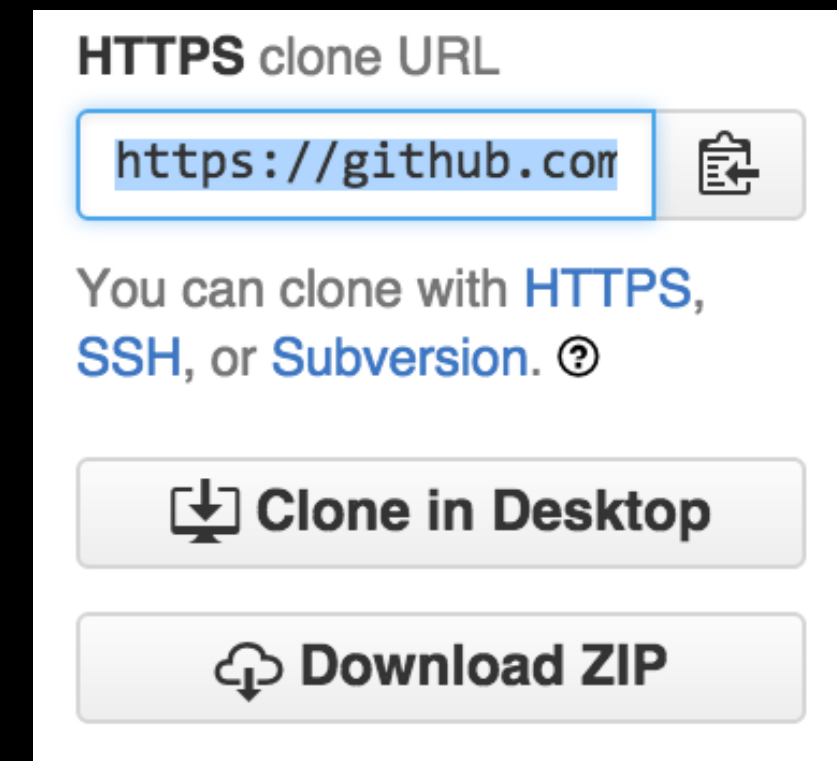
Tübingen - Codevember 2015

01 - Einrichtung und erster Start



Erstmaliger Clone vom Projekt,
Beispiel mit HTTPS und clone URL:

Rechts unten den Link kopieren und
Github Accountinformationen
bereithalten...



Das Repo ist verfügbar auf: <https://github.com/p-schulz/codevember2015>

01 - Einrichtung und erster Start

```
Schulz@Macbook911:~/Documents$ mkdir codevember2015
```

```
Schulz@Macbook911:~/Documents$ cd codevember2015/
```

```
Schulz@Macbook911:~/Documents/codevember2015$ git clone https://github.com/p-schulz/codev
```

```
Schulz@Macbook911:~/Documents/codevember2015$ ll
```

```
total 16
```

```
drwxr-xr-x  4 Schulz  staff   136B  16 Nov 12:36 .  
drwx-----+ 31 Schulz  staff   1,0K  16 Nov 13:28 ..  
-rw-r--r--@  1 Schulz  staff   6,0K  16 Nov 13:27 .DS_Store  
drwxr-xr-x  7 Schulz  staff   238B  16 Nov 13:20 codevember2015
```

```
Schulz@Macbook911:~/Documents/codevember2015/codevember2015$ ll
```

```
total 32
```

```
drwxr-xr-x  7 Schulz  staff   238B  16 Nov 13:20 .  
drwxr-xr-x  4 Schulz  staff   136B  16 Nov 12:36 ..  
-rw-r--r--@  1 Schulz  staff   6,0K  16 Nov 12:36 .DS_Store  
drwxr-xr-x 15 Schulz  staff   510B  16 Nov 13:21 .git  
-rw-r--r--  1 Schulz  staff   116B  16 Nov 13:20 .gitignore  
-rw-r--r--  1 Schulz  staff   748B  16 Nov 11:42 README.md  
drwxr-xr-x 10 Schulz  staff   340B  16 Nov 13:26 codevember15_
```

Ordner erstellen und das Projekt mit Git auf den lokalen Rechner klonen.

01 - Einrichtung und erster Start

```
Schulz@Macbook911:~/Documents/codevember2015/codevember2015/codevember15$ ll
total 128
drwxr-xr-x  10 Schulz  staff   340B  16 Nov 13:26 .
drwxr-xr-x   7 Schulz  staff   238B  16 Nov 13:20 ..
-rw-r--r--@  1 Schulz  staff   8,0K  16 Nov 13:12 .DS_Store
-rw-r--r--   1 Schulz  staff   17K  16 Nov 13:12 Assembly-CSharp-Editor.csproj
-rw-r--r--   1 Schulz  staff   21K  16 Nov 13:12 Assembly-CSharp-firstpass.csproj
drwxr-xr-x   9 Schulz  staff   306B  16 Nov 13:14 Assets
drwxr-xr-x  23 Schulz  staff   782B  16 Nov 13:26 Library
drwxr-xr-x  19 Schulz  staff   646B  16 Nov 13:26 ProjectSettings
-rw-r--r--   1 Schulz  staff   1,9K  16 Nov 13:26 codevember15.sln
-rw-r--r--   1 Schulz  staff   568B  16 Nov 13:10 codevember15.userprefs
```

- Struktur des Projektverzeichnis -

Bitte beachtet, dass der Library-Ordner nie committed und gepushed werden sollte. Auf der nächsten Folie wird darauf eingegangen wie das mithilfe der .gitignore Datei verhindert wird.

Nach Importieren von neuen Inhalten (Bilder, Skripte etc.) sind diese im Assets-Ordner verfügbar (wird auch noch später erläutert).

01 - Einrichtung und erster Start

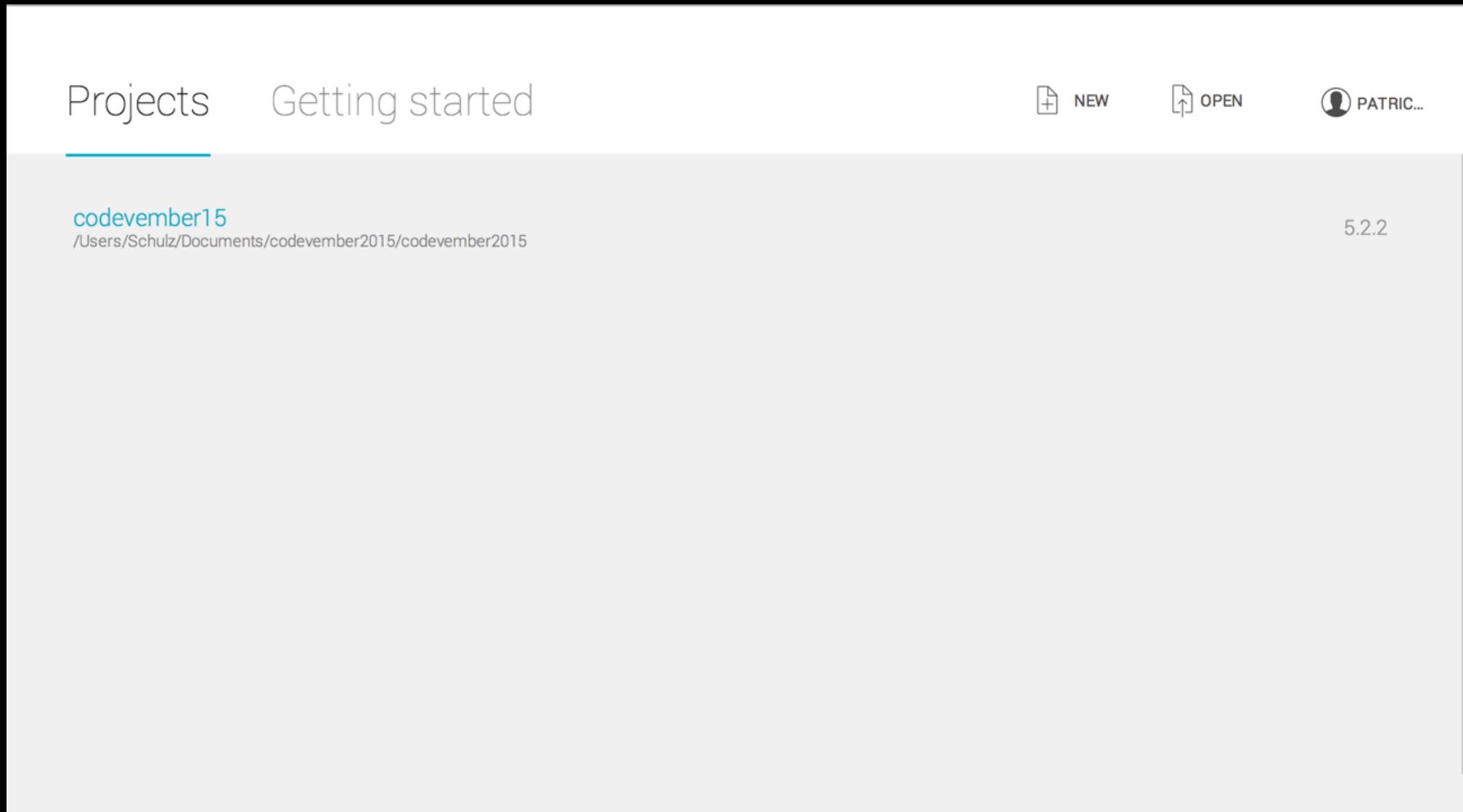
```
codevember15/Temp/  
codevember15/Library/  
codevember/Obj/
```

```
*.csproj  
*.unityproj  
*.sln  
*.user  
*.userprefs  
*.DS_store
```

Die Datei .gitignore sollte unbedingt in das Git-Wurzelverzeichnis, da sonst beim Pushen und Pullen unnötige Bibliotheks- und Metafiles transferiert werden.

Falls nicht vorhanden selbst erstellen und obigen Text einfügen. Das Projekt selbst wurde mit der Option erstellt, die Metafiles nur lokal zu speichern und für Versionsverwaltung nicht zwingend notwendig im Projektordner zu behalten.

01 - Einrichtung und erster Start



← Erster Programmstart

Die nächsten Folie beziehen sich auf die Benutzung der „Indie-Version“. Sollte aber auch kein Problem sein, da wahrscheinlich niemand Geld für Unity ausgeben wird ;-)

Nach Installation und Login oben auf „OPEN“ klicken und den Ordner „codevember15“ im Git-Verzeichnis auswählen. Das Projekt sollte nun in der Liste stehen und auch beim nächsten Start auswählbar sein.

01 - Einrichtung und erster Start

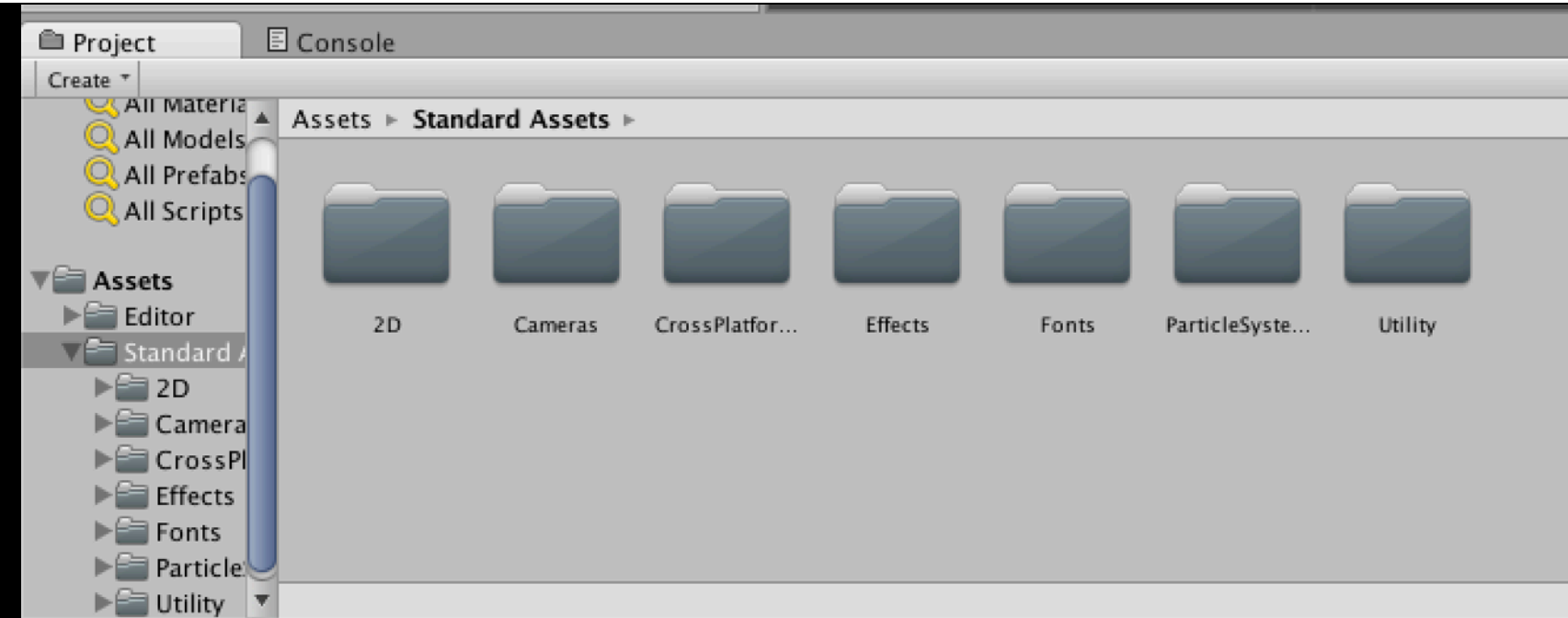
Ansicht nach Programmstart:

Das große Mittelfeld enthält für uns nur zwei wichtige Tabs - Scene und Game. Die Scene ist unsere Standardansicht in der wir die aktuelle Szene gestalten können. Unter Game können wir uns eine Live Version ansehen und einen Eindruck bekommen, was den Anwender wirklich erwartet.

Daneben existiert links die Projekthierarchie, in der alle Objekte der Szene aufgelistet sind. Wird ein Objekt angewählt, so erscheinen rechts im Inspector alle Eigenschaften und Parameter dieses Objekts. Zuletzt sehen wir unten unsere gesamte Ordnerstruktur und den Tab für die Konsole.

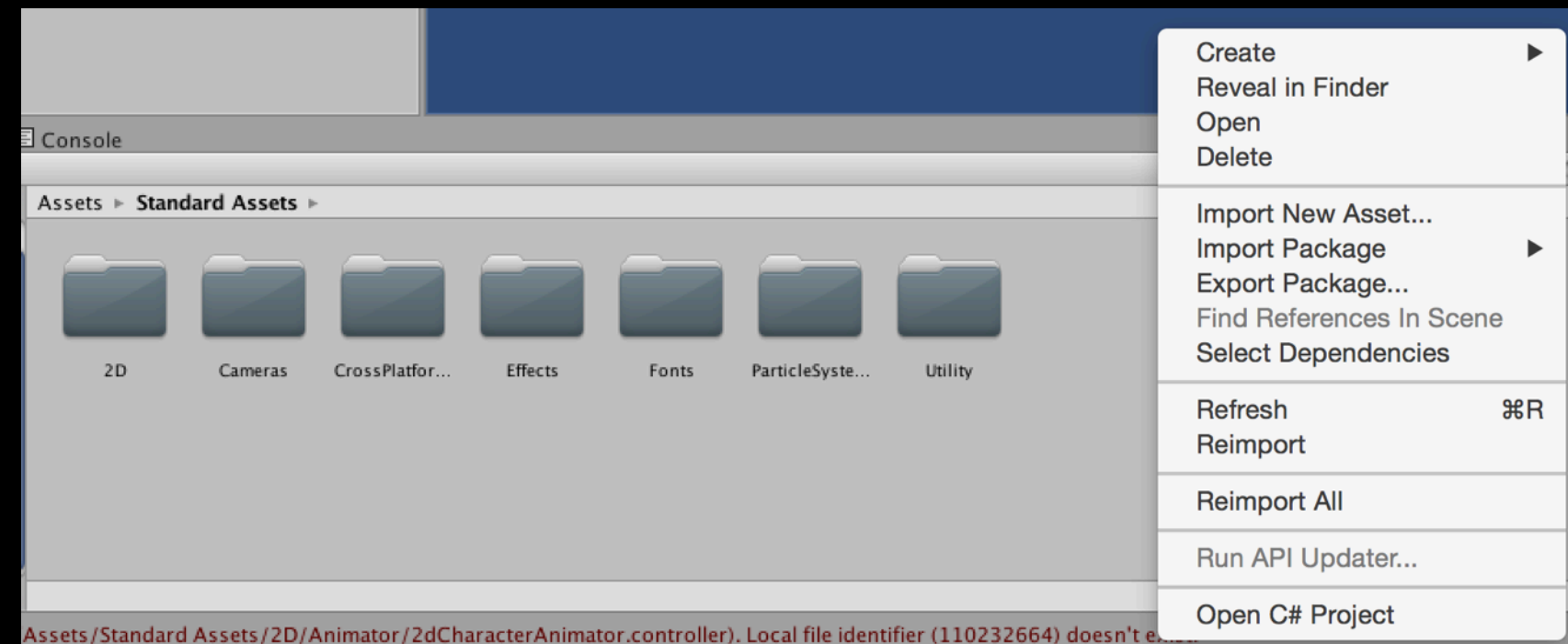
02 - Die Programmoberfläche: Projektstruktur

Im Projektverzeichnis finden wir alle Dateien die in unserem Projekt zur Verfügung stehen.



Rechte Maustaste

Über einen Klick auf die rechte Maustaste in einem Ordner können neue Objekte oder Assets Erstellt oder Importiert werden.

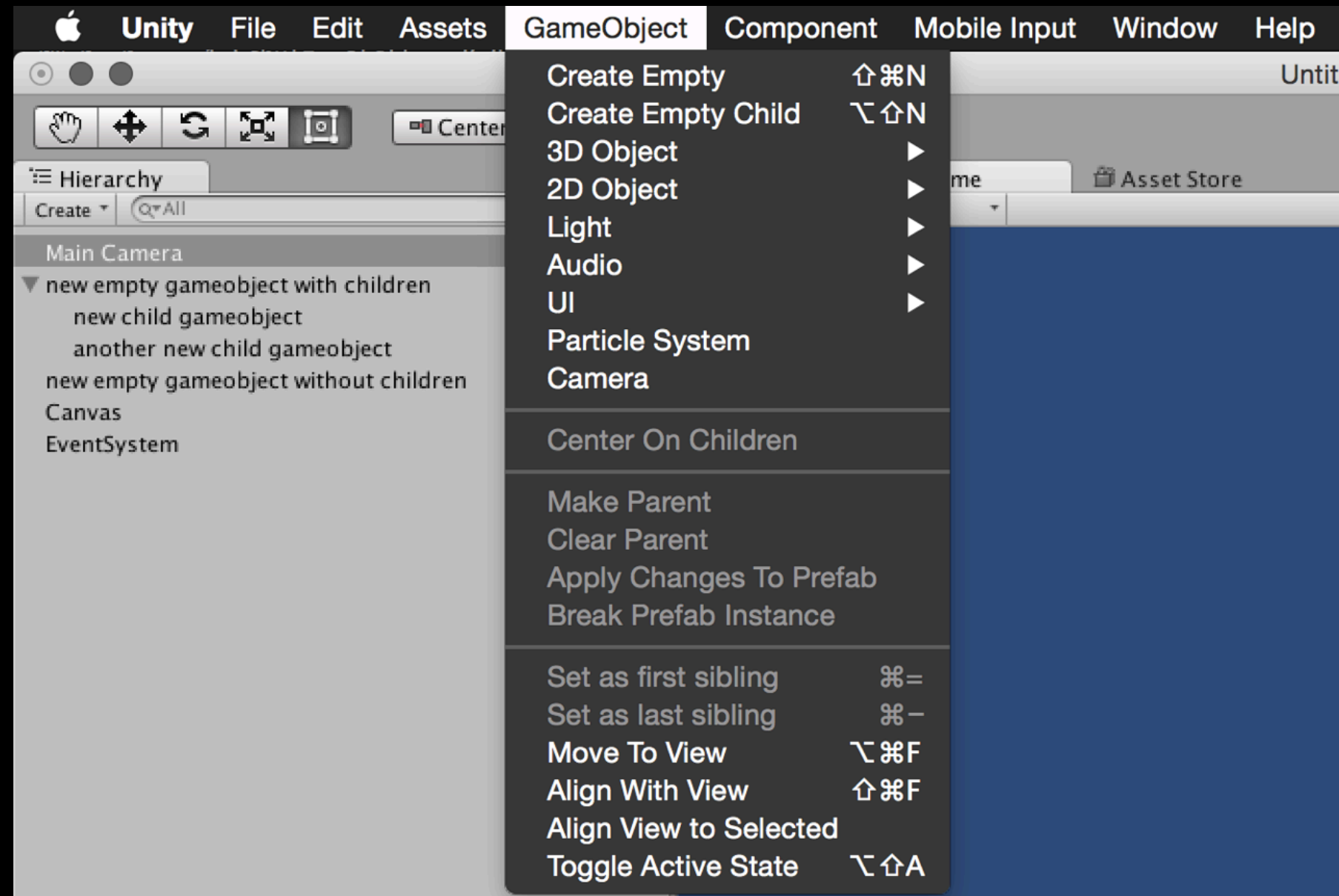
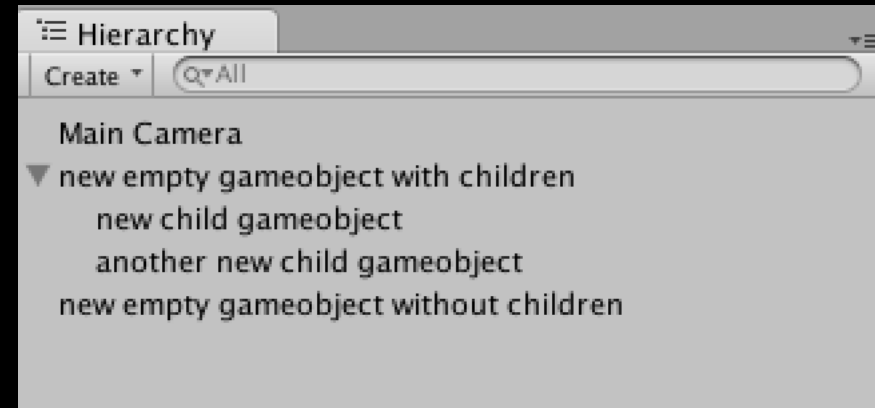


02 - Die Hierarchie und ihre Objekte

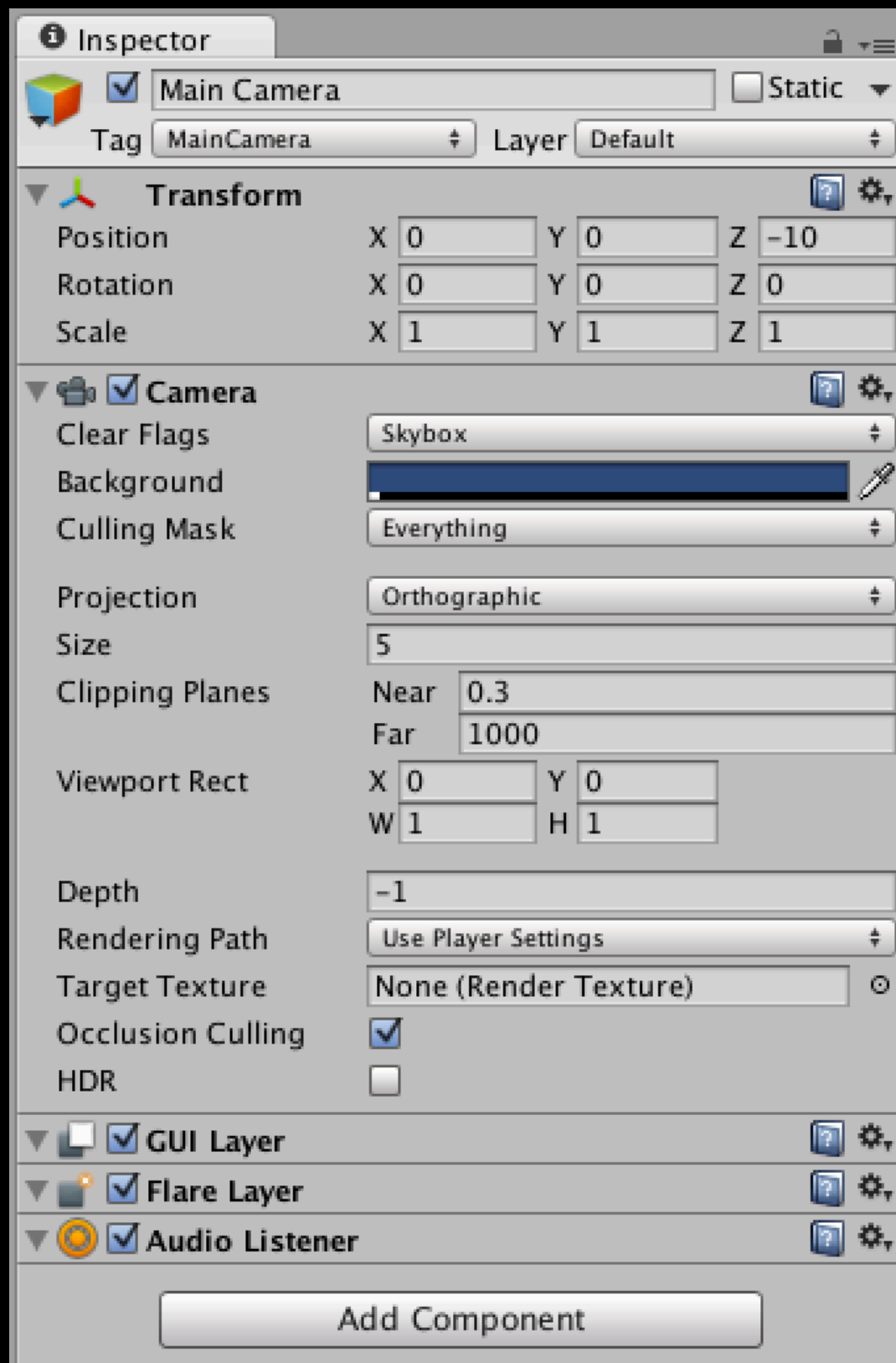
In der Hierarchie sind alle Objekte der aktuellen Szene aufgelistet. Deaktivierte GameObjects sind ausgegraut.

Zur Erstellung neuer Objekte kann man die rechte Maustaste oder den Reiter „GameObject“ in der Menüleiste benutzen.

Gameobjects dürfen (und sollen in einigen Fällen) auch leer bleiben. Kind-Objekte unterhalb eines GameObjects können dadurch zusammengefasst ausgewählt werden um einige Funktionen zu erleichtern.



02 - Der Inspector



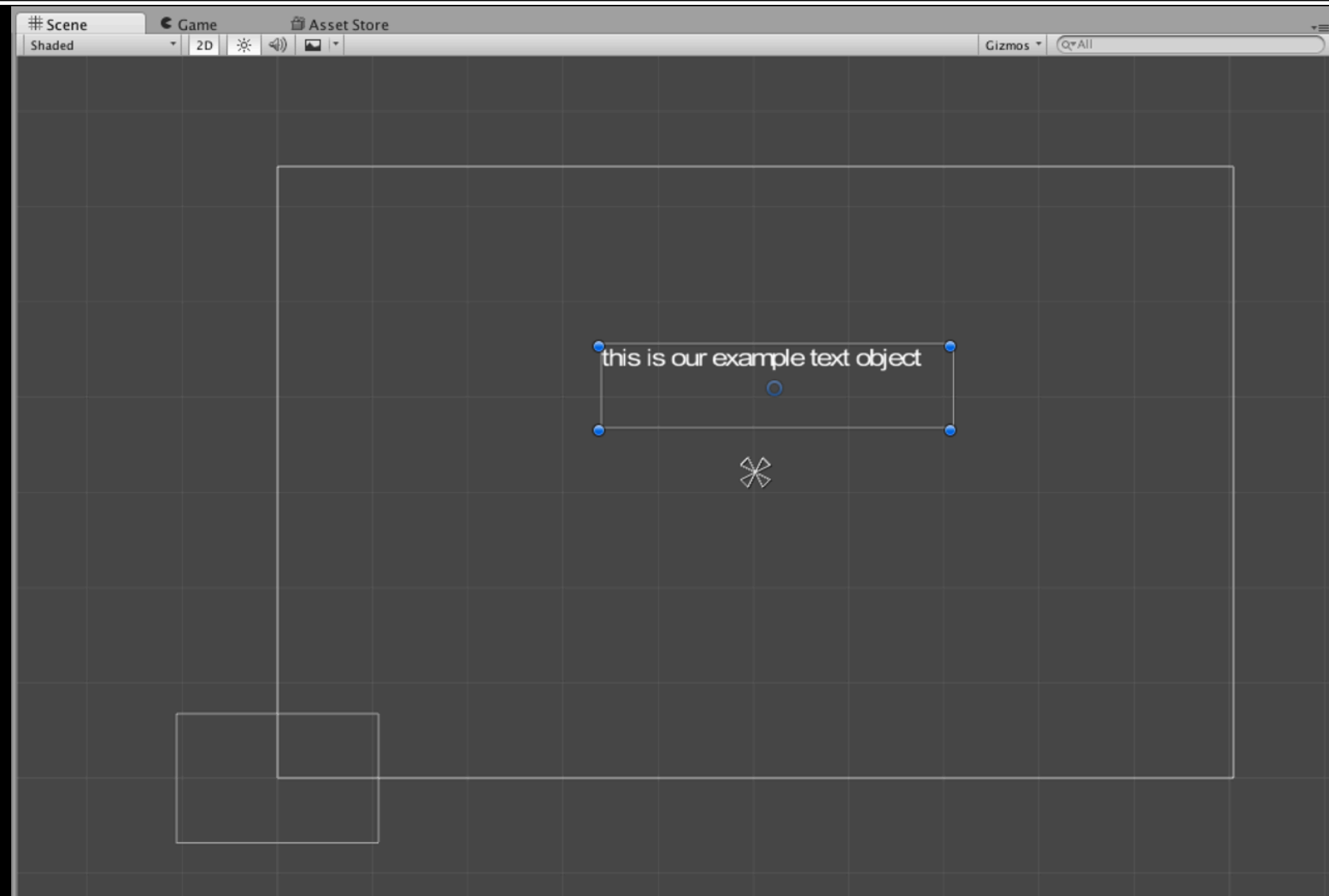
In diesem Beispiel wurde das Gameobject „Camera“ in der Hierarchie ausgewählt. Nun können alle Eigenschaften (z.B. Position und Ausrichtung) dieses Objekts geprüft und verändert werden. Lasst euch nicht von der Fülle an Informationen abschrecken! ;-)

Später gehe ich noch im Detail darauf ein, wie man einem Gameobject Komponenten (Hitboxes, Skripte, Effekte usw.) hinzufügen kann und warum dieses Konzept essentiell für Unity-Projekte ist.

02 - Unsere Scene-View

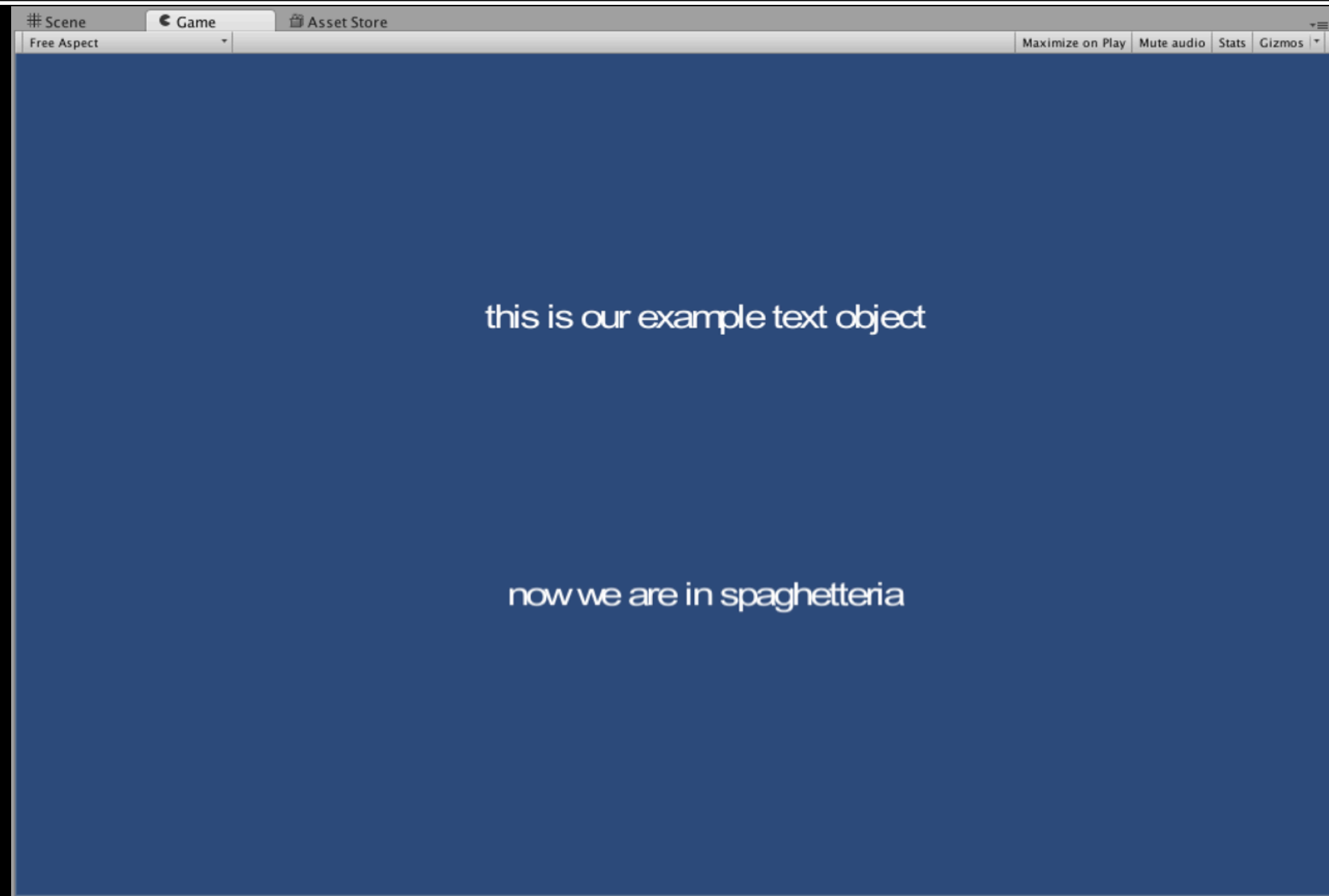
Alle Gameobjects (bis auf ein paar wenige besondere) sind in der Scene sichtbar. In unserem 2D Projekt können sie leicht herumgeschoben werden.

Unser Projekt beinhaltet zu Beginn schon einen Canvas (kleines Rechteck unten links) und eine Camera (Sichtbereich ist das große Rechteck). Alles was sichtbar ist und innerhalb des Sichtbereichs der Kamera liegt wird später gerendert.



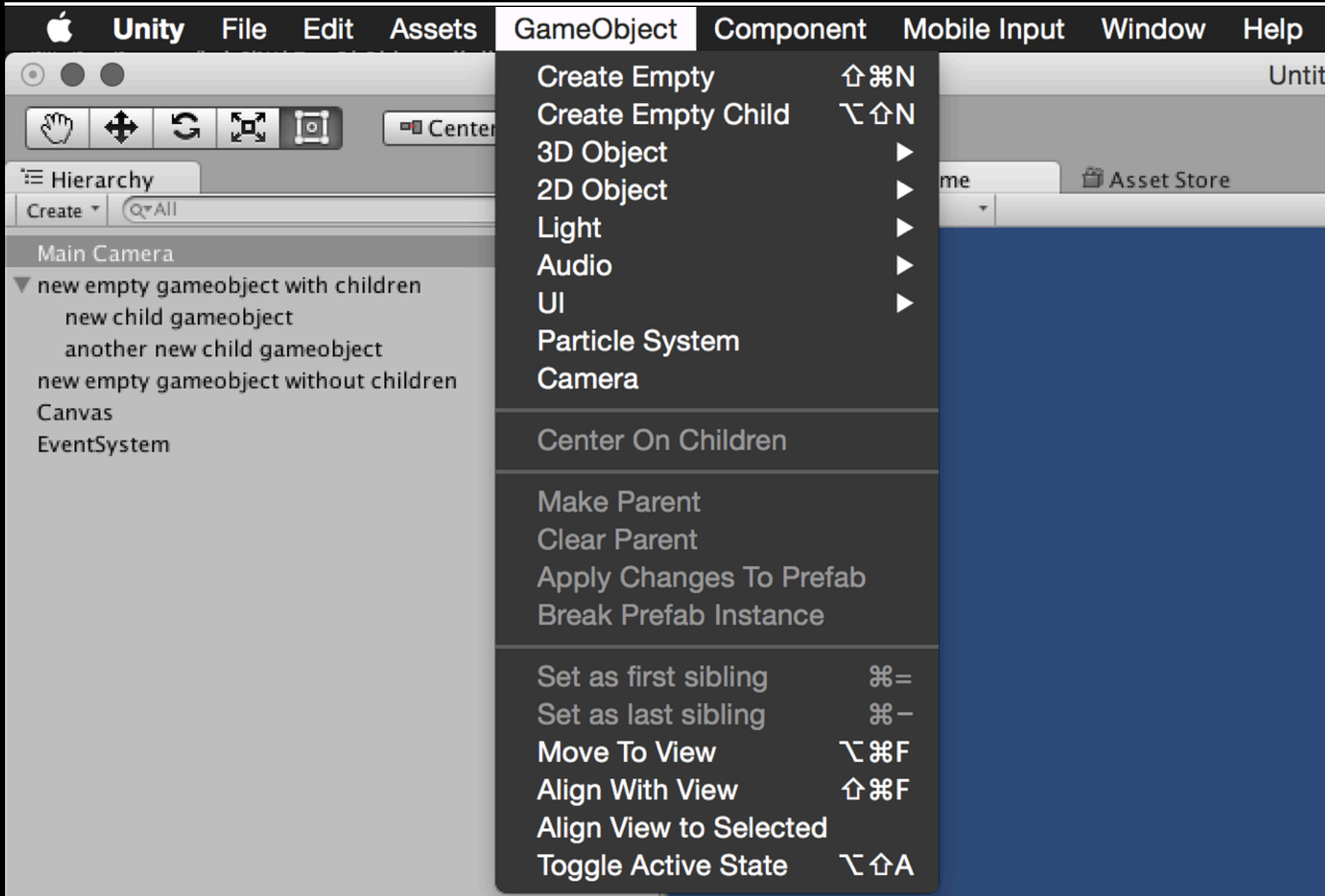
02 - Die Game-View

Wie schon vorhin kurz erwähnt kann das Ergebnis jederzeit in der Game-View betrachtet werden. Hat bis hier her alles bei euch geklappt, solltet ihr in etwa so was hier sehen →



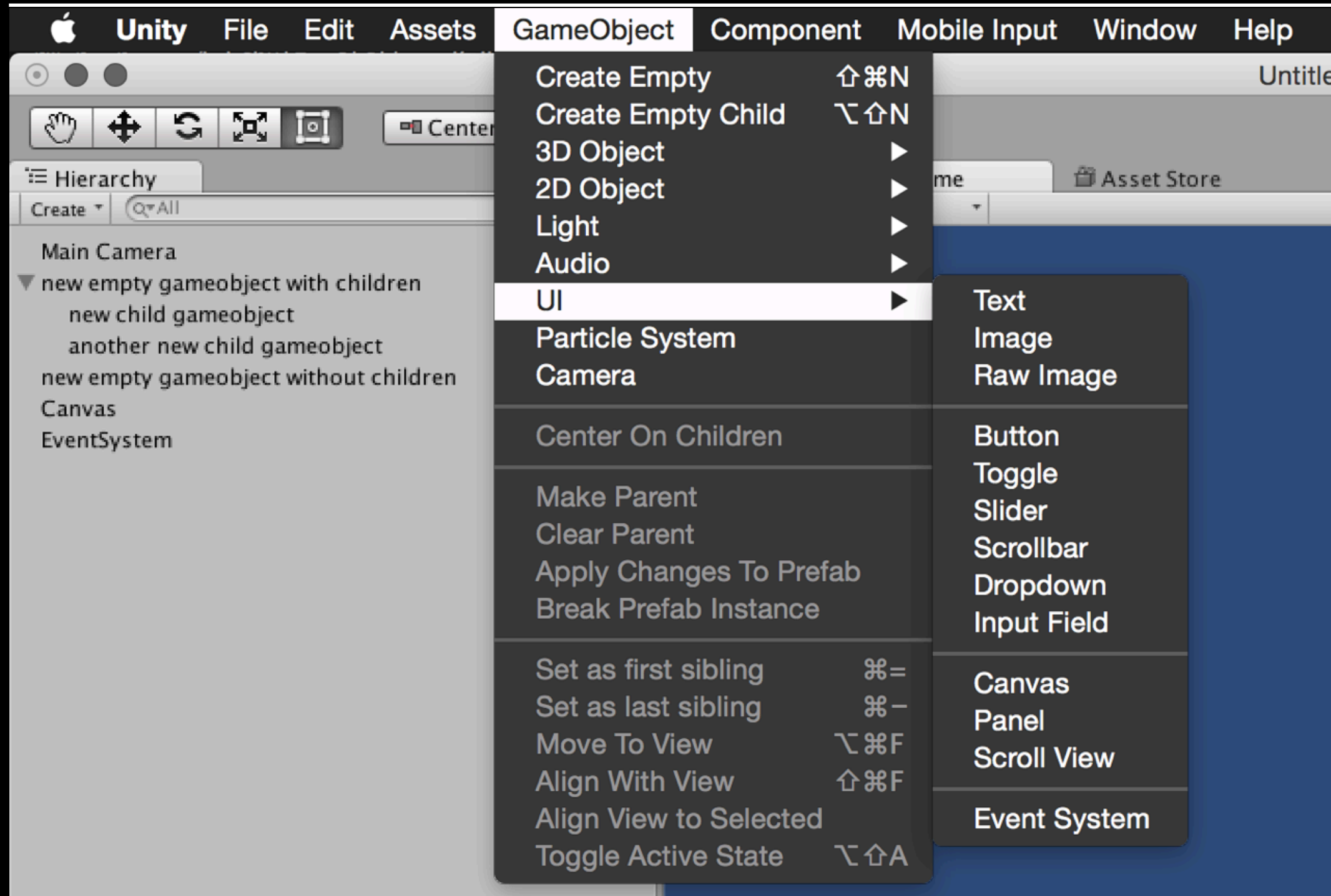
Details zur Game-View werden bei Bedarf noch Live erklärt.

03 - GameObjects erstellen



Gameobjects könnt ihr mithilfe der Menüs in der Menüleiste oder über das Menü mit der rechten Maustaste erstellen. Zunächst interessieren uns nur leere und UI-Gameobjects. Über UI kann zum Beispiel ein Canvas, sprich „Leinwand“, erstellt werden. Darauf können wir später unsere Szene detailreich gestalten.

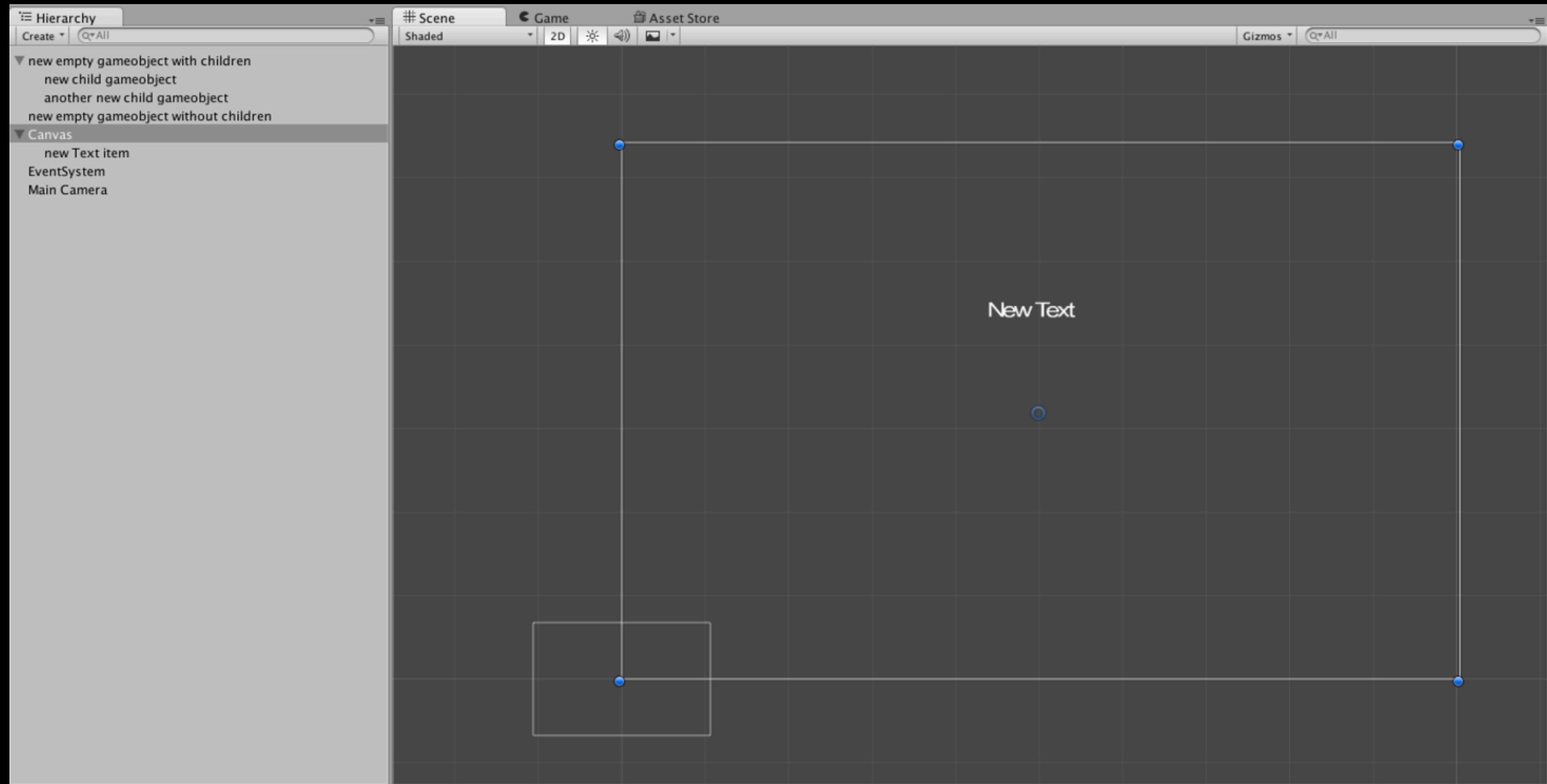
03 - GameObjects Inhalt geben



An dieser Stelle ist es als Übung erwünscht, dass ihr versucht Bilder und Buttons zu erstellen, und euch diese Objekte im Inspector anschaut. Überlegt euch nachdem ihr die Datentypen und Strukturen von Unity kennt, was für Komponenten man diesen Objekten hinzufügen könnte und wie sie dann eingesetzt werden.

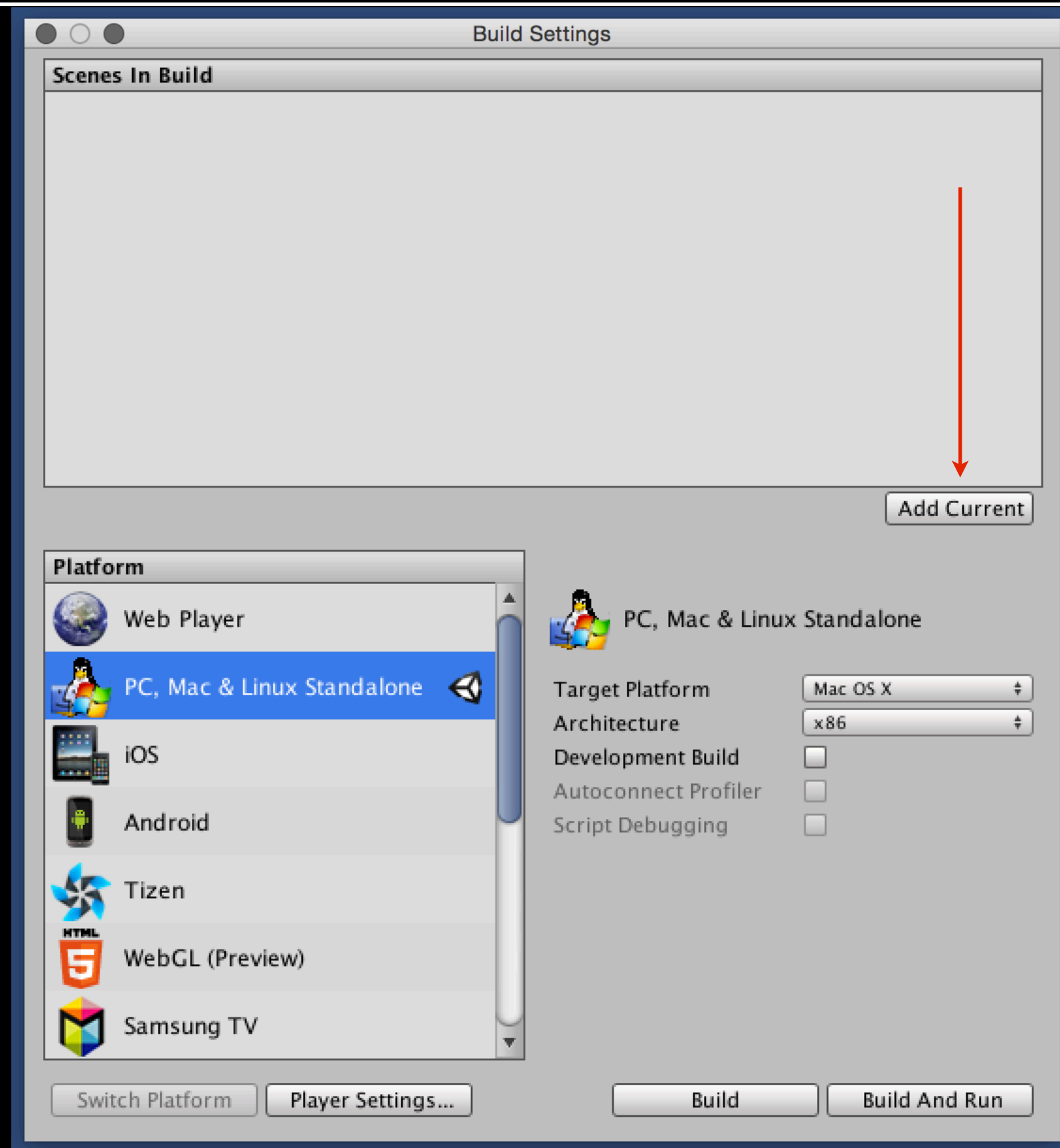
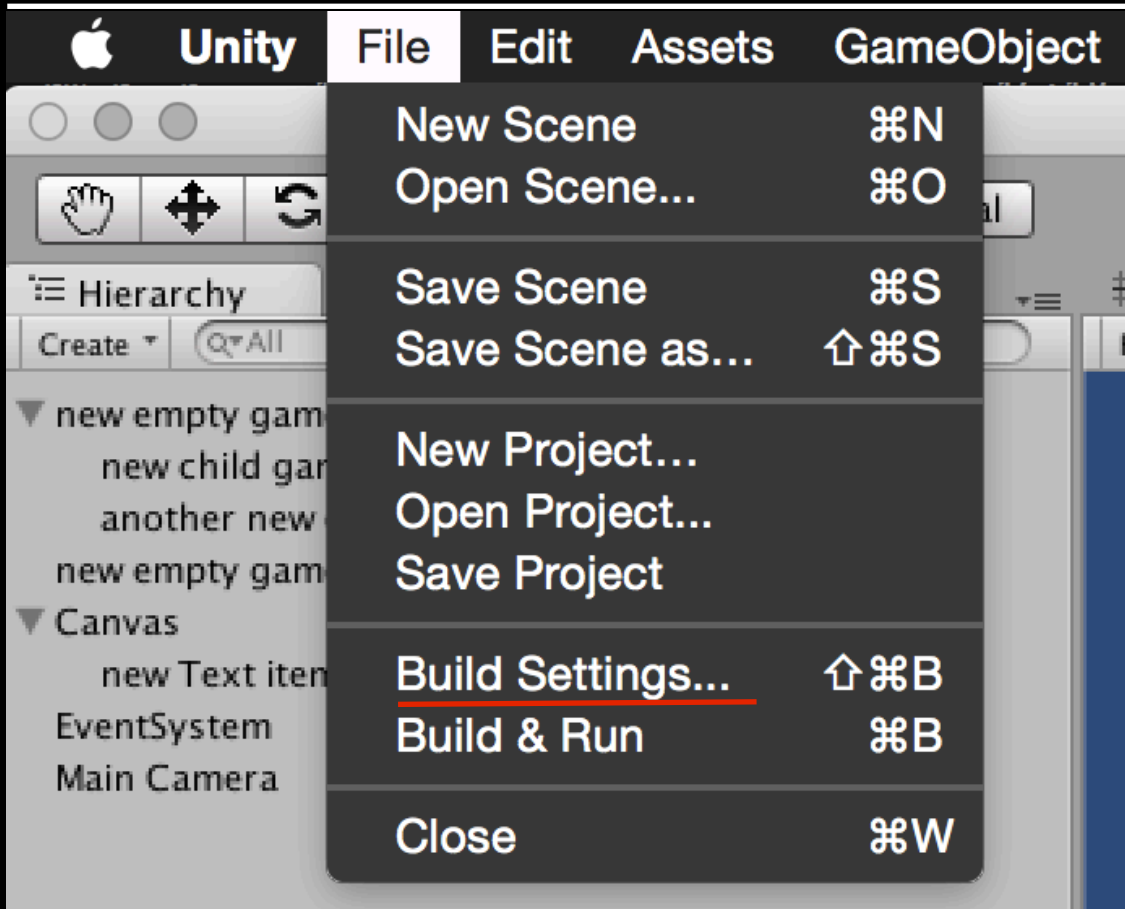
Habt ihr schon die eingerückten Elemente bemerkt? Versucht doch mal die Objekte in der Liste zu vertauschen und zu verschachteln.

03 - Die Szene gestalten



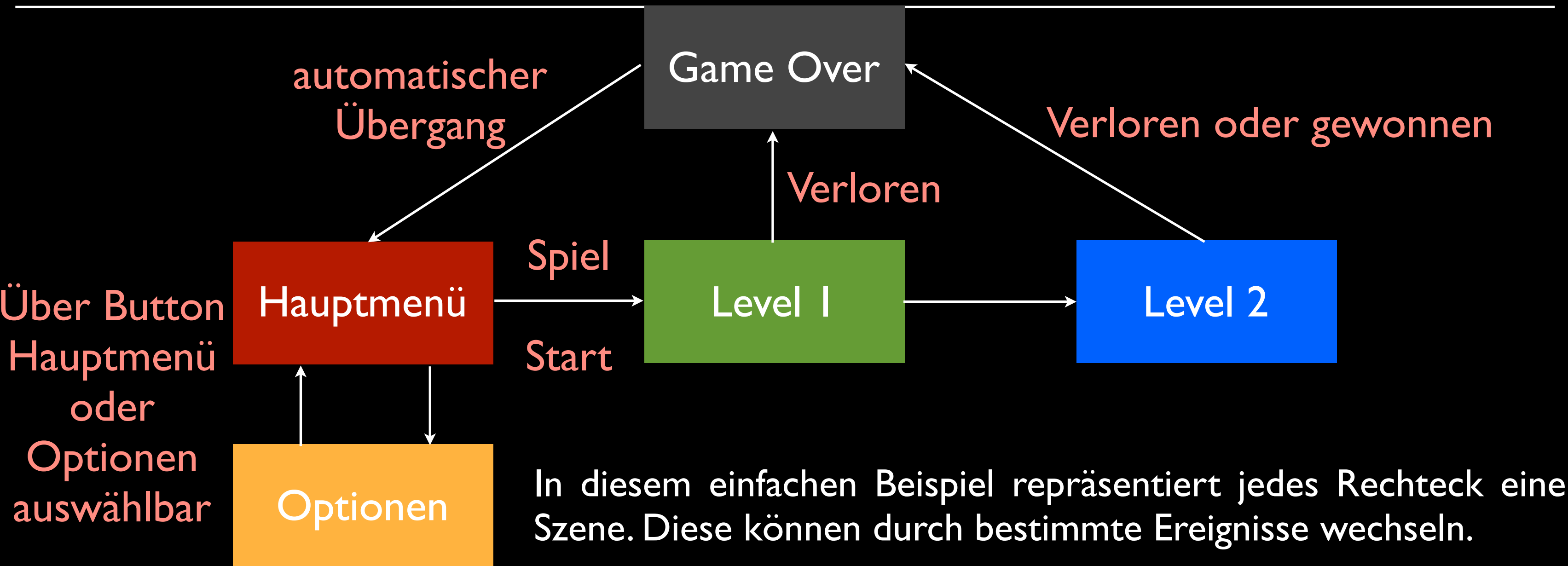
Objekte in der Szene können einfach über die Hierarchie oder das entsprechende Blueprint ausgewählt werden. In der Scene-View könnt ihr es dann einfach verschieben und neu positionieren. Fügt als Übung neue Text-Objekte ein und stellt sie schön dar!

04 - Allgemeine Projekteinstellungen



Damit der Build später auch funktioniert, müssen die Szenen die später verfügbar sein sollen auch in den Build integriert werden. Dazu geht ihr auf die Build-Settings und fügt die aktuelle Szene zu (ist bereits hinzugefügt im Startprojekt).

05 - Das Szenen-Konzept



Ereignisse werden ausgelöst, wenn ein bestimmtes Skript darauf wartet, dass sich z.B. der Szenen- oder Spielstatus verändert und eine entsprechende Funktion aufruft. Aber Achtung: Nicht für jede kleine Veränderung sofort eine neue Szene erstellen!

Spätestens jetzt müsst ihr wohl oder übel einen kleinen Blick in die Dokumentation von Unity werfen (<http://docs.unity3d.com/ScriptReference/>).

06 - Unity Datentypen verstehen

Die verwendete Sprache wird C# sein. Erfahrungen mit verwandten Sprachen wie C, C++ oder Java sind von Vorteil da die Syntax C-ähnlich ist, allerdings nicht Voraussetzung.

Skripte können direkt neu erstellt und bearbeitet werden (Projektstruktur, Create New: Script). Vorteil von C# ist dabei, dass jedes erstellte Skript eine eigene Klasse darstellt, und jede Klasse alle anderen verfügbaren Klassen von Beginn an sieht. Das Schwierige jetzt:

Wie spreche ich Gameobjects an und was kann ich damit machen?!

Dafür bietet uns Unity die hauseigene Klasse GameObject.

06 - Unity Datentypen verstehen

C# ist heftig objektorientiert, also hoffe ich dass ihr objektorientiert coden könnt (diejenigen von euch die Coden wollen). Als kurze Erinnerung:

Gameobjects haben Attribute:

- Ist das Objekt aktiv?
- Ist das Objekt statisch?
- Wo liegt unser GameObject gerade? (Transform)

Gameobjects haben Public-Funktionen:

- Aktiv oder Inaktiv setzen
- Komponenten hinzufügen oder entfernen

Gameobjects haben statische Funktionen:

- Objekt finden, klonen, zerstören, bei Szenenwechsel nicht zerstören etc.

06 - Unity Datentypen verstehen

Bitte schaut euch in der Dokumentation ein paar weitere Klassen an die interessant sein könnten. Hier ein paar Anregungen ;-)

Animation

Audio

(Box)Collider

Canvas

Camera

Event

GUI

Input

Material

Object

Particle

Physics2D

Rect :D

Screen

Sprite

Vector2

Ich hoffe diese kleine Einführung macht euch den Einstieg einfacher und hat ein klein bisschen was gebracht. Fröhliches Coden!