# Notizen Kapitel 2: Crafting Scenes and Game Elements

***Erste Begriffe***

**Scene:** Spiel aufgeteilt in Scenes, sodass jeweils nicht das gesamte Spiel im RAM sein muss, sondern nur die jeweilige Scene

**GameObject:** Jede Scene enthält ihre GameObjects, z.B. den Spieler, Gegner, Gegenstände, etc.

**Views:** = Ansichten im Unity Editor, z.B. Scene view, Game view, Hierarchy view, Project view, Console view, Inspector view

***Neue Scene und erstes GameObject***

1. Nach Start des neuen Unity Projects gibt es bereits eine default scene. Wir kreieren aber eine eigene → Menü File → New Scene → Basic URP → Create
2. Einfügen eines GameObjects → Menü GameObject → 3D Object → Cube

**Gray-boxing:** Wir benutzen einfache GameObjects, um ein Spiel zu entwickeln. Das ermöglicht einfaches Testen. Cube ist ein sehr einfaches GameObject.

***Durch die Scene view navigieren***

Für die Navigation gibt es mehrere Möglichkeiten

- First-person view: navigieren wie ein Spieler während einem Game.
  **Halte rechte Maustaste ständig gedrückt!**
  - Maus-Bewegung steuert Blickrichtung der Kamera an ihrer Position
  - Pfeiltasten bewegen Kameraposition horizontal (mit Shift geht's schneller)
  - Tasten QE bewegen Kameraposition vertikal
- Object view: auf ein GameObject fokussieren.
  **Wähle das Ojekt in der Hierarchy view an.**
  - Taste F → Fokus auf das Objekt.
  - Alt + Maus → um das Objekt kreisen mit Blick auf das Objekt

***GameObjects manipulieren***

Objekt anwählen und dann das Transform-Tool verwenden → **Transform Gizmo** erscheint.

- Objekt verschieben → bewege die **farbigen Pfeile**.
- Objekt insgesamt skalieren → bewege den **grauen Würfel** im Zentrum.
- Objekt in Einzelrichtung skalieren → bewege **farbige Würfel** aussen.
- Objekt rotieren → bewege **farbige Ringe**.

**Achtung!** Es gibt einen Unterschied zwischen lokalen und globalen Koordinaten. Aufgrund von Drehungen können die Koordinatensysteme nicht mehr in Übereinstimmung sein. Und das ist auch gut so. Wir wollen diese Möglichkeit haben, denn z.B. wollen wir einfach längs der allgemeinen Koordinaten Verschiebungen vornehmen können, aber wir möchten auch Verschiebungen längs der lokalen Koordinatenachsen leicht handhaben können.
**Rechts oben wird stets die Ausrichtung der globalen Koordinatenachsen gezeigt.**

***GameObjects und Komponenten***

GameObjects setzen sich aus **Komponenten** zusammen. Unity bietet viele Komponenten an, z.B. das Abspielen eines Sounds, das Erzeugen eines «Gegenstandsgitters» (**mesh**), die Gestaltung der Oberfläche des Objekts oder physikalische Eigenschaften des Objekts.
Man könnte meinen, in der **Inspector view** rechts werden einfach die Eigenschaften eines GameObjects aufgeführt und können dort editiert werden. Aber das ist so nicht ganz richtig.

Die dort vorhandenen Werte und Einstellungen gehören zunächst einfach zu einer Komponente und das Zusammenspiel mehrerer Komponenten macht dann das GameObject aus. Z.B. kreiert der **Mesh Renderer** das durch den **Mesh Filter** vorgegebene 3D-Modell unseres Cubes und zwar am Ort und in der Grösse und Ausrichtung, die von der **Transform**-Komponente vorgegeben werden. Die **Box Collider**-Komponente schliesslich verleiht dem Objekt eine physikalische Form, die für Berührungen resp. Kollisionen von Bedeutung ist.

**Beispiel:** Wir machen aus dem Cube eine Sphere:

- Wähle im Mesh Filter die **Sphere**
- Die Sphere soll ein massiger Körper sein → Add Component: **Physics / Rigidbody**
- Sinnvoll: Umbenennung des Objektes, am einfachsten im Inspector
- Wenn wir das Spiel laufen lassen und die Sphere auf eine schiefe Ebene fallen lassen, so hüpft sie darauf komisch herum. Grund: Sie sieht nur aus wie eine Sphere. Ihr physikalischer Körper ist aber immer noch ein Cube → **Box Collider entfernen** (Remove Component) und stattdessen **Sphere Collider** als Komponente hinzufügen.

***Verständnis für Objekt-Hierarchie***

- Unterordnen von Objekten passiert in der Hierarchy view
  → unterzuordnendes Objekt auf übergeordnetes Objekt ziehen.
  Das untergeordnete Objekt gehört nun zum übergeordneten Objekt.
- Manchmal ist ein **Empty** GameObject hilfreich. Ein solches beinhaltet nur die Transform-Komponente. Wofür braucht man das?
  - Zusammenfassen mehrerer Objekte zu sinnvoller Gruppe, manchmal auch nur um die Übersicht in der Hierarchy view zu verbessern.
    **Achtung!** Nicht übertreiben damit → ab mehr als zwei Ebenen kann sich die Game Performance verschlechtern.
  - Bei vielen Objekten ist der Mittelpunkt auch der **Pivot** (= Drehpunkt). Dies verunmöglicht die Drehung um einen anderen Pivot. Der Pivot des Empty GameObjects kann hingegen frei festgelegt werden (Pivot-Ansicht). Ist nun das eigentliche GameObject dem EmptyObject untergeordnet, so kann der Drehpunkt mit dem Empty GameObject frei festgelegt werden.

***Prefabs***

- **Prefab** = Selbst hergestellte Vorlage für GameObject → ziehe GameObject aus Hierarchy view in Project view → Prefab erscheint dort als neues Asset
- Erstellen einer neuen Instanz (= Ausgabe) eines bestimmten Prefabs
  → ziehe Prefab aus Assets in Scene
- Modifikation eines Prefabs, sodass alle Instanzen betroffen sind
  → Doppelklick auf den Prefab in Project view startet **Prefab Edit Mode**
  → Änderungen vornehmen. Edit Mode durch Klick auf **Scenes** oben links verlassen.
- Bestehende GameObjects können einem existierenden Prefab zugeordnet werden
  → Doppelklick auf das Object in Hierarchy view, dann **Prefab / Replace…**
- Soll sich eine Instanz von den andern unterscheiden, so können wir sie anwählen und in der Inspector view verändern. Solche Veränderungen an der einzelnen Instanz übersteuern (**override**) die Eigenschaften des Prefabs oder ergänzen diese. Overrides können leicht überblickt/geändert werden → Overrides Drop Down im Inspector.
- Auch Untervarianten von Prefabs können erzeugt werden → Instanz eines Prefabs unter neuem Namen nochmals in die Project view ziehen.