Beschreibung UML-Diagramm

Aufbauend auf dem Sequenzdiagramm, ist im Folgenden das UML-Diagramm zu dem Unity-Projekt unserer Gruppe. Hierbei sind Elemente der Unity-Engine nicht dargestellt, da diese sowohl keine Eigenleistung der Gruppe ist als auch in C und C++ geschrieben ist und daher dem klassischen Modell der Objektorientierung nicht folgt. Eine Darstellung würde der Präsentation unseres Projektes mittels eines UML-Diagramms schaden. Trotzdem ist die folgende Darstellung des UML-Diagramms von diesem Faktum beeinflusst, worauf unter Anderem in dieser Beschreibung eingegangen wird.

Die Entwicklung in Unity ist Objektorientiert aufgebau. Es erben alle Klassen von der Klasse „Object". Grundsätzlich erben nur zwei Klassen von „Object" und alle weiteren vom User erstellten Klassen teilen sich in „GameObject" und „Component" auf. Um dies darzustellen wurden Operatorblöcke benutzt. Diese dienen nur der Übersichtlichkeit und haben keine weitere Funktion. Zusätzlich wurden vor den Bezeichner der Klassen jeweils ein Zeichen mit einem Buchstaben und einer Farbe platziert. Analog zu den Operatorblöcken dienen diese Zeichen der Übersichtlichkeit.

Generell werden Klassen bzw. Objekte die von „GameObject" erben, in einer sogenannten Szene dargestellt. Eine Szene ist ein Objekt der Unity Engine und stellt den vom Nutzer der Software sichtbaren Bereich da. Wie im UML-Diagramm zu sehen, haben Klassen, die von „GameObject" erben fast ausschließlich „Components" als Felder. Diese Beschreiben die Verhaltenseigenschaften der Instanz des „GameObjects". „Components" werden im Inspector dargestellt, der nur in der Entwickleroberfläche sichtbar ist. Sie enthalten meistens grundlegende Datentypen, die als Parameter für die Unity Engine dienen. Hat zum Beispiel die Instanz von „Player1" ein „Component" „rigidbody2D", welcher wiederum mit „mass" = 1.5 instanziiert wird, dann wird die Unity Engine die Instanz von „Player1" mit dem Faktor 1.5 in Richtung des Gravitationspunkt beschleunigen.

Zusätzlich zu den von Unity vorgegeben „Components" (Klassen die von „Component" erben und ein orangenes Zeichen haben), können eigene „Components" in Form von „Scipts" erstellt werden. Diese erben alle von „MonoBehaviour" welche wiederum von „Behaviour" erbt. Sie können also wie andere „Components" einem „GameObject" zugewiesen werden, und bieten so die Möglichkeit, Verhalten von z.B. „Player1" genauer und „freier" zu beschreiben.

In der im UML-Diagramm als Klasse dargestellten Legende sind die Framework-Eigenen Bezeichner für die Eigenschaften public und private von Klassenattributen- und –methoden zu finden.