C# Code Richtlinien

Allgemein:
Verwendung der üblichen C# Namenskonventionen und Quellcode Gliederung, sowie englischer Namensgebung für Quellcode und deutsche Kommentare.

Namespaces:
Verwendung von PascalCasing und Namen passend zu den Verzeichnissen im Projekt mit Ausnahme des Assets Ordners.

Klassen-, Structnamen:
Verwendung von PascalCasing und gegebenenfalls Aussagekräftigen Akronymen.
Es sollten keine "_" in Klassennamen verwendet werden.

Attribute:
Verwendung von PascalCasing und gegebenenfalls Aussagekräftigen Akronymen.
Es sollten keine "_" in Namen verwendet werden.
Attribute sollten am Anfang der Klasse stehen und nach Zugriffsmodifizierer absteigend sortiert werden (public, internal, protected, private)
Konstanten werden normal in CAPS dargestellt.
Attribute werden standardmäßig im Konstruktor und nicht direkt initialisiert.
Implementierung als Propertys ist zu bevorzugen. Dabei kann fuer Get und Set jeweils eigene zugriffsmodifizierer festgelegt werden. Beispiele:

private string Test { get; set; } **(get private; set private)**
public Vector3 { get; private set } **(get public; set private)**

Wenn die Get und Set Methoden noch zusaetzliche Befehle benoetigen, oder fuer die Variablen default Werte gesetzt werden müssen, dann verwendet man folgendes Pattern:

```
private readonly string test = "test";
public string TEST
{
  get
  {
    return test;
  }

  set
  {
    Test = value; // Value ist eine Variable die bei einer Set Zuweisung den Wert darstellt der
                  zugewisen werden soll.
  }
}
```

Dabei wird ein Privates Attribut mit camelCase und eine Property mit ueblichem PascalCasing Deklariert. Dann koennen auch in den Methoden weitere Befehle eingefuegt werden.

Methoden:
Verwendung von PascalCasing und gegebenenfalls Aussagekräftigen Akronymen.
Es sollten keine "_" in Namen verwendet werden.
Methoden sollen nach Zugriffsmodifizierer absteigend sortiert werden (public, internal, protected, private)
Reihenfolge der Methodeneigenschaften:

[Zugriffmodifizierer][Modifizierer][Return][Name][Generics][Parameter][Andere Einschränkungen]

Private static string getValue()
Public abstract int calculate<TValue>(Tvalue a)
Public sealed double calculate(float a)

Generics beginnen normal mit einem T und einem aussagekräftigen Namen z.B. **Tkey, Tvalue,** etc.

Parameter verwenden camelCase. Für referenzparameter (C++ Referenzen mit &) wird das Schlüsselwort **out** vor den Parameter geschrieben.

Public bool TryCalculating(int a, int b, out int c)

Parameter mit defaultwert stehen am ende und müssen nicht zwingend übergeben werden.

public static bool StoreString(string name, string content, bool overwrite = false)

Lokale Variablen werden mit camelCasing benannt.

Interfaces:
Selbe Namensgebung für Methoden wie bei Klassen/Strukturen.
Der Interfacename beginnt jedoch immer mit einem großen "i".

IDisposable, ICompareable, ICollection etc.

Exceptions und Logs:
Allgemein sollten Exceptions mit try/catch Bloecken zur Fehlerbehandlung verwendet werden und nicht Fehlercodes und Returnwerte.
Beschreiben der Fehler und Logtexten auf Deutsch.

Unity Objekte

Benennung in Englisch und verwendung von PascalCase.
"_" und Sonderzeichen sollten nach moeglichkeit vermieden werden.

Git Richtlinien

Allgemein:
Eintragen von Summary/Beschreibung auf Deutsch.

Summary:
Eintragen der Tasks mit #Nummer und Name die von dem Commit/Push betroffen sind.

#34 Regale modellieren # 37 Ladungsträger modellieren usw.

Description:
Stichpunktartig beschreiben was implementiert bzw. geändert wurde.