**GameManager**

*Der GameManager verwaltet die Eingaben der Player und gibt diese an das aktuelle Subgame weiter.*

*Es kann von jeder Stelle des Spiels benutzt werden, d.h. es ist nicht an die Existenz eines GameObjects gebunden.*

* existiert als eine einzige Klasseninstanz (Singleton).
* besitzt eine maximale Spieleranzahl.
* besitzt eine Liste der Namen der Subgame-Prefabs.
* besitzt 1-4 Player.
* kann gestartet werden.
* kann pausiert werden.
* kann nach dem Pausieren fortgesetzt werden.
* kann einem Spieler einen Punkt zuschreiben.
* kann einem Spieler einen Punkt abziehen.
* Kann den Gewinner ermitteln.
* kann neue Player-Spielobjekte instantiieren.
* kann auf die Reaktionsmeldung eines *Players* reagieren.

**SubgameManager**

*Der SubgameManager verwaltet die Subgame-Szene und das darin ablaufende Subgame. Er platziert auf Anweisung des GameManagers Steuerungselemente der Benutzer.*

* besitzt die beiden Bereiche, in denen die Steuerelemente der Spieler platziert werden.
* besitzt ein Subgame.
* kann ein neues Subgame über seinen Namen instantiieren und starten.
* kann prüfen, ob das Subgame zu Ende ist.
* kann die Steuerelemente für die gegebene Anzahl an Spielern platzieren.

**Subgame**

*Ein Subgame ist ein Spielmodus, bzw. eine Phase des Spiels. In Unity bildet jedes Subgame ein eigenes Prefab, das in die Spielfläche der Scene des laufenden Spiels geladen wird.*

* besitzen alle eine gemeinsame Zahl, die beschreibt, auf wie viele ReactionTasks bis zum Fortschreiten zum nächsten Subgame korrekt reagiert werden muss.
* besitzt einen Namen.
* kennt den Namen seiner, die Tasks aufbauenden Objekte enthaltende, XML-Datei.
* besitzt eine knappe Aufgabenbeschreibung.
* besitzt eine Zahl, die beschreibt, auf wie viele ReactionTasks noch korrekt reagiert werden muss.
* kann das aktuelle ReactionTask ersetzen.
* kann mitteilen, ob aktuell auf eine Spielerreaktion gewartet wird.
* kann mitteilen, ob es abgeschlossen (beendet) ist.
* kann pausiert werden.
* kann fortgesetzt werden.
* kann prüfen, ob das aktuelle Task abgelaufen ist.

**TerminableTaskSubgame : Subgame**

*Ein TerminableTaskSubgame ist ein Subgame, das aus zeitlich begrenzten Abschnitten (ReactionTasks) besteht. Während der gesamten Anzeigezeit behält die kritische Eigenschaft des ReactionTasks seinen Zustand bei.*

* kann mitteilen, ob es abgeschlossen (beendet) ist.
* kann prüfen, ob das aktuelle Task abgelaufen ist.

**IndefiniteTaskSubgame : Subgame**

*Ein IndefiniteTaskSubgame ist ein Subgame, das aus zeitlich unbegrenzten ReactionTasks besteht. Seine ReactionTasks bleiben solange angezeigt, bis ein Spieler reagiert. Während der Anzeigezeit des ReactionTasks verändert sich dessen kritische Eigenschaft ein- oder mehrmalig.*

* kann mitteilen, ob es abgeschlossen (beendet) ist.
* kann die angezeigte kritische Eigenschaft auf „gesucht“ oder „nicht gesucht“ ändern.
* kann prüfen, ob das aktuelle Task abgelaufen ist.

**MitarbeiterFotoSubgame : TerminableTaskSubgame**

**Player : MonoBehaviour**

*Ein Player ist das Spielobjekt eines teilnehmenden Spielers. Dieser reagiert über seinen Button auf die einzelnen ReactionTasks und bekommt über einen Text und das Aussehen seines Buttons Rückmeldungen vom GameManager.*

* besitzt einen Typ für den Anzeigezustand seines Buttons.
* besitzt einen Button.
* besitzt ein Anzeigefeld für Nachrichten an den Benutzer.
* besitzt eine Punktezahl die als Standard addiert/subtrahiert wird.
* besitzt Erfolgsnachrichten.
* besitzt Misserfolgsnachrichten.
* besitzt eine aktuelle Punktezahl.

* kann auf das Gedrückt werden des zugeordneten Buttons reagieren.
* kann das Ergebnis seiner Reaktion entgegennehmen.
* kann eine Meldung an den Benutzer ausgeben.
* kann das Aussehen des User Interfaces zurücksetzen.
* kann seinen Punktestand ändern.
* kann das Aussehen seines Buttons auf einen Zustand anpassen.

**MenuConfigManager : MonoBehaviour**

* besitzt eine Liste von SubgameToggle-Objekten, die Informationen über das Subgame und die angezeigte Auswahlmöglichkeit enthalten.
* besitzt eine Referenz auf die Fläche, auf der die Toggles platziert werden sollen.
* besitzt eine Referenz auf das Prefab zum Erstellen der Toggles.
* besitzt die Zahl der maximal ohne Scrollen sichtbaren Toggles.
* besitzt den Wert des Ankers der Zielfläche, der beim Platzieren der Toggles angepasst werden muss.
* kann auf das Drücken des Hauptmenü-Buttons reagieren.
* kann auf das Drücken des Spielstart-Buttons reagieren.
* kann die Liste verfügbarer Subgames laden.
* kann aus der Liste der Subgames Toggles im Einstellungsmenü platzieren.
* kann die getroffene Auswahl an Subgames speichern.
* kann auf die angezeigten Toggles eine vorher gespeicherte Auswahl anwenden.

**XmlReader**

*Ein XmlReader lädt für Subgames, deren ReactionTasks auf Inhalten aus XML-Dateien basieren, die benötigten Informationen aus der jeweiligen mit dem Subgame verknüpften XML-Datei.*

* existiert genau ein Mal (Singleton).
* kann aus einer Datei Elemente lesen und dabei eine zufällige Kombination der Kindelemente verschiedener Elemente, sowie eine Angabe, ob diese zum selben Elternelement gehören als Strings zurückgeben.
* kann aus einer Datei die Namen der verfügbaren Subgames laden.

**zu XML-Dateien:**

Eine Entität in einer XML-Datei enthält immer 2 Einträge.

Dies kann eine Kombination zweier Kindelemente sein, die zusammen auftretend eine Reaktion korrekt machen. Eine andere Art ist eine Kombination eines Kindelementes und eines einfachen Wahrheitswertes, der angibt, ob eine Reaktion auf das Element korrekt ist.

Es gibt eine XML-Datei, die für jedes bisher implementierte Subgame den Namen seines Prefabs und einen zugehörigen, im Auswahlmenü angezeigten Namen enthält.