# Battleship

Das Battleship Projekt ist eine Platform, die verschiedene KIs für Schiffe-Versenken in einem Turnier gegeneinander antreten lässt. Dabei spielt jeder gegen jeden. Es werden immer 1000 Matches gespielt, um den Gewinner zwischen zwei KIs zu ermitteln. Dies soll eventuelle Zufälle vermeiden und wirklich den besseren Gewinner herausfinden.

Es ist ratsam, während der Entwicklungszeit den eigenen SourceCode vor den anderen Teilnehmern geheim zu halten, damit diese sich nicht auf die eigene Taktik vorbereiten können.

## Aufbau

Das Projekt besteht aus 3 Teilen. Dem GameManager, der BattleSDK und dem KI Projekt.

Der GameManager ist das Herzstück der Platform. Er wird als .exe gestartet, lädt die KIs, lässt sie gegeneinander antreten und ermittelt den Gewinner.

Die BattleSDK ist eine .dll, die dem Teilnehmer ermöglicht, eine eigene KI zu schreiben, die vom System geladen werden kann.

Das KI Projekt wird vom Teilnehmer selbst angelegt. Es handelt sich dabei um eine Klassenbibliothek (.dll). Das Projekt muss die BattleSDK referenzieren, um auf ihre Funktionalität zugreifen zu können.

Wenn die Solution gebaut wird, befinden sich die benötigten Dateien unter battleship/GameManager/bin/Debug:

/

/BattleSDK.dll

/GameManager.exe

/Players/TestKI.dll

/Players/Overkill.dll

Die nicht benötigten KIs können aus dem Players Ordner entfernt werden, da sie sonst mit am Turnier teilnehmen.

## Battle SDK

Die BattleSDK besteht aus 3 einfachen Klassen, die die Integration ins Battleship System erlauben. BattleshipKI, Direction und Ship

### Direction

Direction ist ein Enum, das die Ausrichtung eines Schiffs bestimmt. Ein Schiff kann horizontal oder vertikal auf dem Feld platziert werden.

### Ship

Ein Schiff wird auf dem Spielfeld platziert.

Es hat die Koordinaten X und Y. Diese legen je nach Ausrichtung des Schiffs entweder das oberste oder das ganz linke Feld des Schiffes fest.

Es hat außerdem eine Größe Size. Size ist die Anzahl an Felder, die dieses Schiff einnimmt.

Ein Schiff hat auch eine Direction Dir.

Zuletzt hat jedes Schiff auch eine Liste an Hits. Jeder Hit stellt einen Treffer von der gegnerischen KI dar. Der Integer in der Liste beschreibt den Index des Feldes vom Ursprung aus. Um also von einem Hit auf die absolute Position des Spielfeldes zu kommen ist sowohl die Position des Schiffes als auch dessen Ausrichtung notwendig.

### BattleshipKI

Diese abstrakte Klasse wird vom Teilnehmer genutzt. Der Teilnehmer erstellt genau eine Klasse, die von BattleshipKI erbt.

#### public BattleshipKI(int size);

Der Konstruktor der Klasse. Size ist die Spielfeldgröße. Size = 10 bedeutet also, dass auf einem 10x10 Feld gespielt wird.

Size kann auch -1 sein. In dem Fall wird kein Spiel gespielt, aber das System braucht eine Instanz der Klasse, um GetName() aufzurufen.

#### public abstract void SetShips(List<Ship> ships);

Bevor das Spiel beginnt, muss jede KI ihre Schiffe auf dem Spielfeld platzieren. Dafür wird diese Methode aufgerufen und eine Liste der Schiffe wird mit übergeben. Aufgabe der KI ist es nun X, Y und Dir jedes Schiffes zu setzen. Schiffe dürfen sich nicht überlappen. Es ist nicht erlaubt, Schiffe aus der Liste zu entfernen oder weitere hinzuzufügen. Nachdem SetShips beendet wurde, ist es auch nicht mehr möglich, die Schiffe zu bearbeiten.

#### public abstract String GetName();

Der zurückgegebene String wird vom System dafür verwendet, die KIs im Turnier anzuzeigen.

#### public abstract void Shoot(out int x, out int y);

Wenn die KI am Zug ist, wird Shoot aufgerufen. Die Aufgabe der KI ist es nun, x und y auf die Werte zu setzen, auf die sie gerne schießen möchte.

#### public abstract void Notify(int x, int y, bool hit, bool deadly);

Nachdem die KI geschossen hat, wird Notify aufgerufen. Notify enthält die Informationen, die der Schuss ergeben hat. Neben den Koordinaten ist das die Information, ob ein Schiff getroffen wurde und wenn ja, ob der Schuss das Schiff versenkt hat. Anhand dieser Informationen kann sich die KI ein Bild über die Situation machen.

## GameManager

Um das Spiel etwas zu modifizieren, können im GameManager einige Einstellungen verändert werden. Dies kann nützlich sein, wenn man in der Entwicklung der eigenen KI ist. So können Änderungen schnell getestet werden und die KI auf verschiedene Szenarien vorbereitet werden.

### Program

#### private static int numGames;

Die Anzahl der Spiele, die zwei KIs gegeneinander antreten. Kann währen der Entwicklung auf einen niedrigeren Wert gesetzt werden.

### GameManager

#### private int size;

Die Spielfeldgröße

#### private int shipAmount;

Die Anzahl der Schiffe, die jeder Spieler bekommt