Schiffe versenken

Fachinformatiker Anwendungsentwicklung
E2FI1 2017/2018

Tobias Schacherbauer, Jürgen Huber
Abgabe: 04.07.2018

Inhaltsverzeichnis

Selbstständigkeitserklärung

Hiermit erklären wir, dass wir die vorliegende Projektarbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt haben.

Die Stelle der Projektarbeit, die andere Quellen im Wortlaut oder dem Sinn nach entnommen wurden sind durch Angaben der Herkunft kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.

Ulm, den 04.07.2018

Tobias Schacherbauer, Jürgen Huber

Quellen

Newtonsoft JSON Bibliothek: Serialisierung der Einstellungen ins JSON-Format, um sie auf der Festplatte speichern zu können, Deserialisierung der Einstellungs-Datei, um gespeicherte Einstellungen zu laden

Extended WPF Toolkit: Verwendung des „Color-Pickers“ in den Einstellungen

Projektidee / Projektziel

Das Ziel unseres Projekts ist, ein auf WPF basierendes „Schiffe versenken“-Spiel in C# zu entwickeln. Das Spiel besitzt eine grafische Oberfläche bestehend aus einem Hauptmenü, einem Einstellungsfenster und dem eigentlichen Hauptfenster, welches die zweidimensionalen Spielfelder enthält.

Unser Spiel ermöglicht es dem Benutzer, über das Einstellungsfenster Spielparameter, wie beispielsweise die Spielfeldgröße oder den Spielmodus, anzupassen und in Form einer JSON-Datei zu speichern. Diese wird beim erneuten Programmstart geladen und ausgewertet.

Das Spielprinzip folgt den normalen Regeln eines „Schiffe versenken“ Spiels. Unser Programm ermöglicht es dem Spieler gegen einen KI-Gegner anzutreten, welcher, basierend auf der gewählten Stärke des Gegners, zufällige Felder anvisiert, bis ein Treffer erfolgt.

Anschließend sucht die KI nach den restlichen, von diesem Schiff belegten Feldern. Sobald das Schiff zerstört wurde, geht die KI wieder in den Suchmodus über und visiert zufällig ausgewählte, noch nicht getroffene Felder an.

Sobald alle Schiffe eines Spielers (des menschlichen, oder des KI-Gegners) zerstört wurden, wird der Gewinner in der GUI angezeigt und es erscheint ein Button, welcher den Benutzer zurück ins Hauptmenü führt.

Pflichtenheft

Das Ziel des Projekts ist es, ein Programm zu erstellen, welches das Spielprinzip „Schiffe versenken“ auf einer zweidimensionalen Spielfläche darstellt. Das Spiel ist für einen Spieler ausgelegt, der gegen einen computergesteuerten Gegner (im folgenden KI-Gegner) antritt.

Musskriterien

- Ein grafische Oberfläche mit: Schiffsarten, aktuellem Spielbrett, Trefferanzeige auf dem Spielbrett, Statuskarte mit Anzeige von Treffern und Nicht-Treffern, Zeitanzeige, verbleibende Anzahl an gegnerischen Schiffen
- Der Spieler kann vor dem Spielbeginn seine Schiffe auf dem Spielbrett platzieren
- Vor dem Spielbeginn gibt der Spieler einen Namen an
- Unterschiedliche Spielmodi
    - Standard (klassisches Schiffe versenken)
    - Survival (neue Gegner erscheinen, bis man verliert)
    - Advanced (Survival) Mode (Schiffe besitzen Spezialfähigkeiten, bspw. Torpedo; Kombinierbar mit Survival)
- Einstellungsmöglichkeiten
    - Größe des Spielfelds
    - Schiffsfarben
    - Stärke der KI-Gegner (basierend auf Algorithmus):
        - Leicht: KI-Gegner schießt jedes Mal auf ein zufälliges Feld
        - Normal: KI-Gegner schießt zunächst zufällig und bei Treffer versucht es das getroffene Schiff komplett zu zerstören
    - Anzahl der Schiffe pro Schiffstyp
    - Spielmodus
    - Rundentimer

Wunschkriterien

- KI-Gegner Stärke „Schwer“ (Allwissender KI-Gegner)
- KI-Gegner basierend auf einem Neural Network
- Spezialfähigkeiten für Schiffe:

    - Torpedo (Schuss der eine komplette Spalte trifft)
    - Radar (deckt eine bestimmte Anzahl an Feldern auf)
    - Scattershot (trifft mehrere nebeneinander liegende Felder)
    - Rüstung (gepanzertes Feld (eines Schiffes) muss mehrmals getroffen werden)
    - Reparatur (eine bestimme Anzahl an Feldern kann nach einem Treffer wiederhergestellt werden)
    - Und mögliche weitere
- Es soll eine Auswahl an verschiedenen Munitionstypen geben. Besondere Munition soll begrenzt zur Verfügung stehen
- Nach einem Spiel werden Punkte errechnet und mit dem Spielernamen abgespeichert. Diese Highscore-Liste kann im Spielmenü abgerufen werden
- Es sollen verschiedene Kartentypen zur Verfügung stehen, die alle spezielle Eigenschaften haben, z.B. Hindernisse auf der Karte

Projektverlauf

Allgemeines

Mittwoch, 30.05.:

- Einrichtung eines gemeinsamen Git-Repositories
- Planung des Klassendiagramms (UML)
- Genehmigung der Projektidee

Mittwoch, 06.06.:

- Abgabe des Pflichtenhefts
- Erstellung der geplanten Klassen
- Erstentwurf eines Schiffplatzierungs-Algorithmus

Mittwoch, 13.06.:

- Erstellung einer rudimentären GUI
- Verfeinerung der Schiffsplatzierungslogik
- Erstellung erster Schnittstellen zwischen GUI und Spiellogik

Mittwoch, 20.06.:

- Entwurf eines Zielalgorithmus für KI-Gegner
- Erstellung des Einstellungsfensters
- Bugfixes

Mittwoch, 27.06.:

- Implementierung der Einstellungsspeicherung per Serialisierung
- Weitere Bugfixes in den KI-Algorithmen
- Beginn der Arbeit an der Dokumentation

Mittwoch, 04.07.:

- Arbeit an der Dokumentation

Implementierung

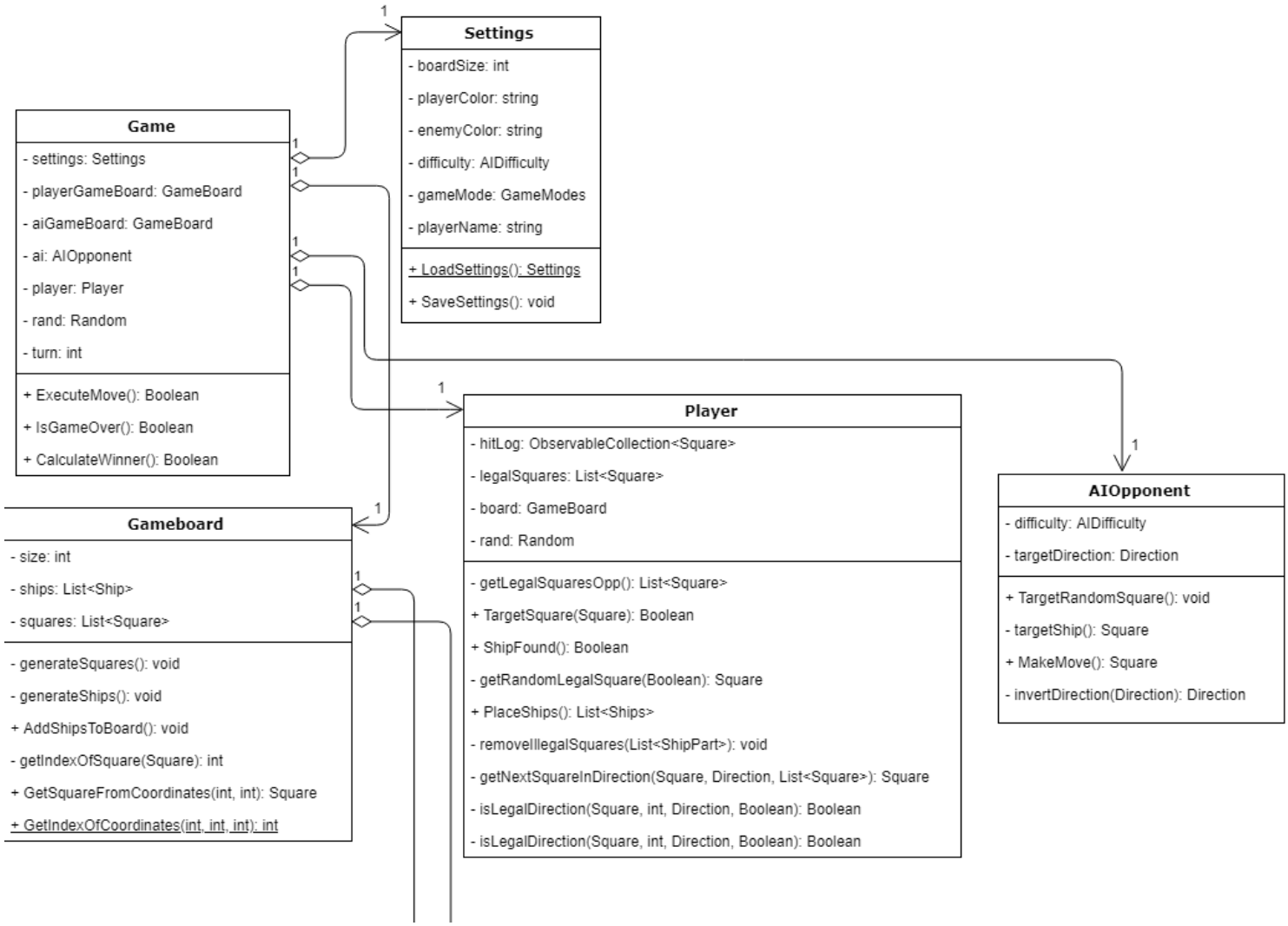
Der erste Schritt bei der Implementierung unseres Projekts war, die vorher im UML-Diagramm geplanten Klassen zu erstellen und zu füllen. Im nächsten Schritt planten wir das Design der GUI und die benötigten Schnittstellen zur Spiellogik, für eine möglichst einfache Wartbarkeit und die bestmögliche Erweiterbarkeit des Programms. Anschließend entwarfen wir Algorithmen, welche das Spielfeld mit

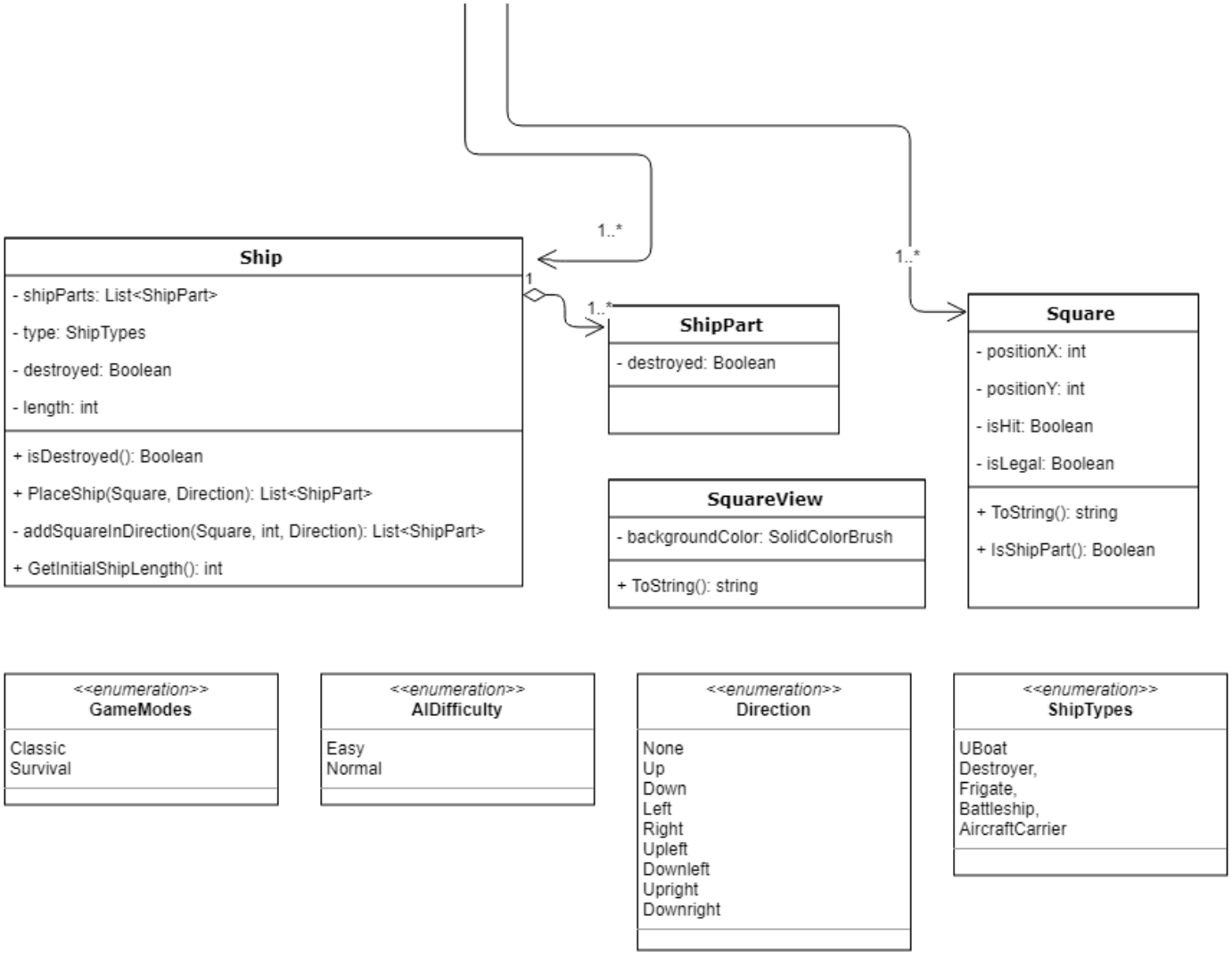
Schiffen füllen und die KI-Gegner befähigt, gefundene Schiffe zielgenau zu zerstören.

Tests

Die gleichzeitige Entwicklung von GUI und Spiellogik ermöglichte es uns, unsere Funktionen und Algorithmen kontinuierlich zu testen, anzupassen und eventuelle Bugs zeitnah zu beheben.

Klassen-Struktur / UML




1..*
1..*
Ship
- shipParts: List<ShipPart>
- type: ShipTypes
- destroyed: Boolean
- length: int
+ isDestroyed(): Boolean
+ PlaceShip(Square, Direction): List<ShipPart>
- addSquareInDirection(Square, int, Direction): List<ShipPart>
+ GetInitialShipLength(): int
1
1..*
ShipPart
- destroyed: Boolean
Square
- positionX: int
- positionY: int
- isHit: Boolean
- isLegal: Boolean
+ ToString(): string
+ IsShipPart(): Boolean
SquareView
- backgroundColor: SolidColorBrush
+ ToString(): string
<<enumeration>>
GameModes
Classic
Survival
<<enumeration>>
AIDifficulty
Easy
Normal
<<enumeration>>
Direction
None
Up
Down
Left
Right
Upleft
Downleft
Upright
Downright
<<enumeration>>
ShipTypes
UBoat
Destroyer,
Frigate,
Battleship,
AircraftCarrier

Bedienung / Spielablauf

Hauptmenü

Das Hauptmenü enthält zwei Buttons, welche es dem Benutzer ermöglichen in das Einstellungsfenster zu wechseln und eine Partie zu starten. Nachdem eine Partie beendet wurde, kann der Benutzer über einen Button ins Hauptmenü zurück gelangen.

Abbildung 1: Hauptmenü

Einstellungsfenster

Im Einstellungsfenster kann der Benutzer Spielparameter verändern. Darunter fallen beispielsweise die Spielfeldgröße, die Spieler, sowie die KI-Gegner Farbe, die Anzahl der Schiffe pro Spieler, sowie der Spielmodus. Über einen Klick auf den „Save“-Button werden diese Einstellungen in einer JSON-Datei gespeichert und auch beim nächsten Programmstart übernommen. Ein Klick auf „Cancel“ führt zurück ins Hauptmenü.

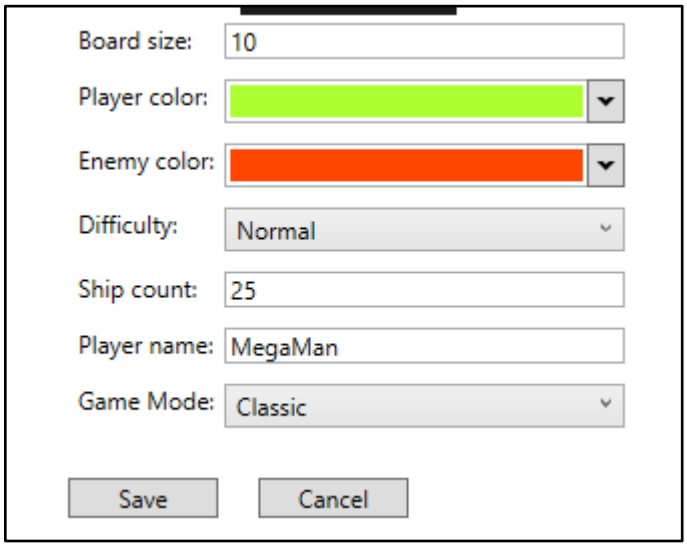


Abbildung 2: Einstellungsfenster

Hauptspiel

Nach einem Klick auf den „Start game“-Button im Hauptmenü gelangt der Benutzer in die Spieloberfläche. Hier werden ihm zwei Spielfelder, sowie zwei ListBox-Elemente angezeigt. Der Schriftzug über den Spielfeldern zeigt dem Benutzer die Zugehörigkeit des Spielfelds, entweder zum Spieler, oder zum KI-Gegner an.

Auf dem Spielfeld des Spielers werden seine Schiffe in den Spielerfarben (auswählbar im Einstellungsfenster) dargestellt. Treffer auf Schiffe des KI-Gegners werden mit dessen Farbe hinterlegt.

Über den Button „Quit game“ hat der Benutzer die Möglichkeit die Partie abzubrechen und in das Hauptmenü zurückzukehren.

Der Spieler und der KI-Gegner führen nun abwechselnd ihre Züge durch. Ein Zug des Spielers besteht aus der Auswahl und dem Doppelklick einer Zelle. Die getroffenen Zellen werden in den ListBox-Elementen aufgeführt.

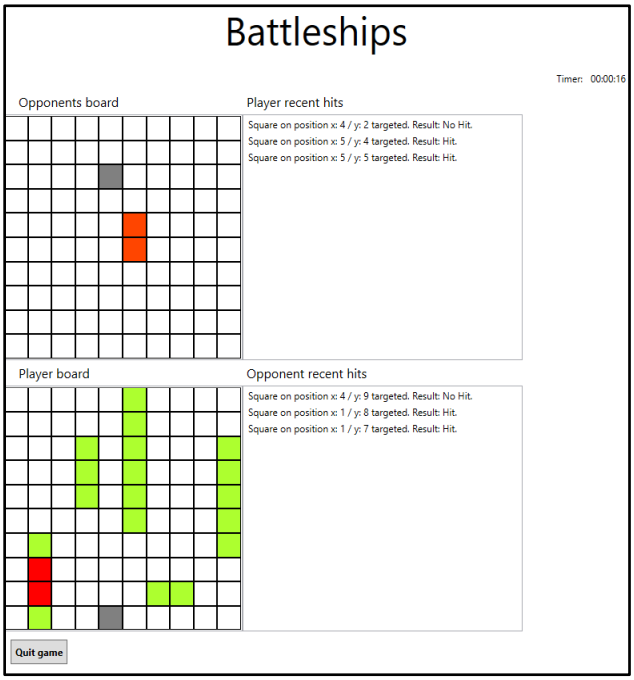


Abbildung 3: Spielfenster

Nachdem alle Schiffe eines Spielers zerstört sind, erscheint ein Schriftzug, welcher den Gewinner der Partie verkündet und den Benutzer über den „Back to start screen“-Button zurück ins Hauptmenü führt. Von dort aus kann eine neue Partie gestartet werden.

Probleme während des Projekts

Durch den erstmaligen Einsatz von WPF in einem C#-Projekt, mussten wir beide erstmal in die gängige Benutzung von Steuerelementen in den XAML-Dateien zurechtkommen. Vor allem die Gestaltung der einzelnen Komponenten (Größen, Hintergrundfarbe) und die Bindung von Eigenschaften an den Programmcode musste verstanden werden. Die Auswahl der richtigen Komponente für das Spielbrett stellte sich auch zuerst als schwerer heraus als erwartet: Es wurde zunächst versucht das Spielbrett mithilfe des DataGrid-Elements zu lösen, was aber nicht möglich war. Die Wahl fiel dann am Ende auf ein mehrspaltiges ListBox-Element.

Probleme traten bei der Testbarkeit der KI-Klasse auf: Hierbei musste vor allem bei Fehlern in der Regel ein komplettes Spiel durchgespielt werden, um so eine Chance zu haben den Fehler in der Logik zu finden. Mithilfe des Debuggers konnten dann erst die Fehlerquellen aufgespürt und behoben werden.

Dadurch, dass wir beide keinen eigenen Laptop zur Verfügung hatten, stellte sich die Entwicklung in der Schule als schwierig heraus. In der Regel musste zu Beginn jeder Stunde auf dem Laptop erst GIT installiert werden und dann der aktuelle Stand geladen werden. Außerdem ist die Visual Studio Version auf diesen Rechnern sehr veraltet und die Entwicklung ist weniger komfortabel, als in der aktuellsten Version.

Fazit

In den Augen der Autoren ist das Projekt insgesamt als Erfolg zu betrachten. Es wurden alle im Pflichtenheft aufgelisteten Musskriterien umgesetzt. Allerdings hat es sich aus Zeitgründen als schwierig herausgestellt, sich mit den Wunschkriterien zu beschäftigen, weshalb diese in der finalen Version nicht umgesetzt wurden. Aufgrund der einfachen Erweiterbarkeit des Programms wird es allerdings keine Probleme bereiten, zukünftig weitere Funktionen und Wunschkriterien einzubauen, bzw. umzusetzen.

Eine nützliche Erweiterung aus den Wunschkriterien des Pflichtenhefts wäre beispielsweise die Addition von einzigartigen Schiffsfähigkeiten, bzw. die Modifikation einzelner Schiffskomponenten. Dies würde das Spiel weiter von der Masse abheben und den Spielspaß beträchtlich steigern.
Außerdem könnte eine weitere KI-Gegner Stufe hinzugefügt werden, welche auf noch komplexeren Algorithmen basiert, um dem Spieler eine noch größere Herausforderung zu bieten.