13.1.2017

Space Shooter Reloaded

Dokumentation – Computertechnik

Torben, Nils

Computer Technik
Dokumentation Halbjahr 1

Gliederung:

1. Einleitung

Wir haben uns entschieden in Ct einen Space Shooter zu programmieren. Man soll mit seinem Raumschiff die Gegner besiegen und den Höchstmöglichen Punktestand erreichen. Die Level sind zeitlich beschränkt und man verliert, wenn man kein Leben mehr hat.
An dem Projekt haben Torben Striegel und Nils Wais gearbeitet.
Wir haben über das Programm Source-Tree und Bitbucket gearbeitet, so konnten wir jederzeit weiter programmieren ohne sich immer absprechen zu müssen damit man nicht die Klassen von dem anderen verändert.
Das Ziel war das Spiel so zu programmieren, dass man einfach neue Raumschiffe und Level einfügen kann ohne dass man viel neues Programmieren muss.

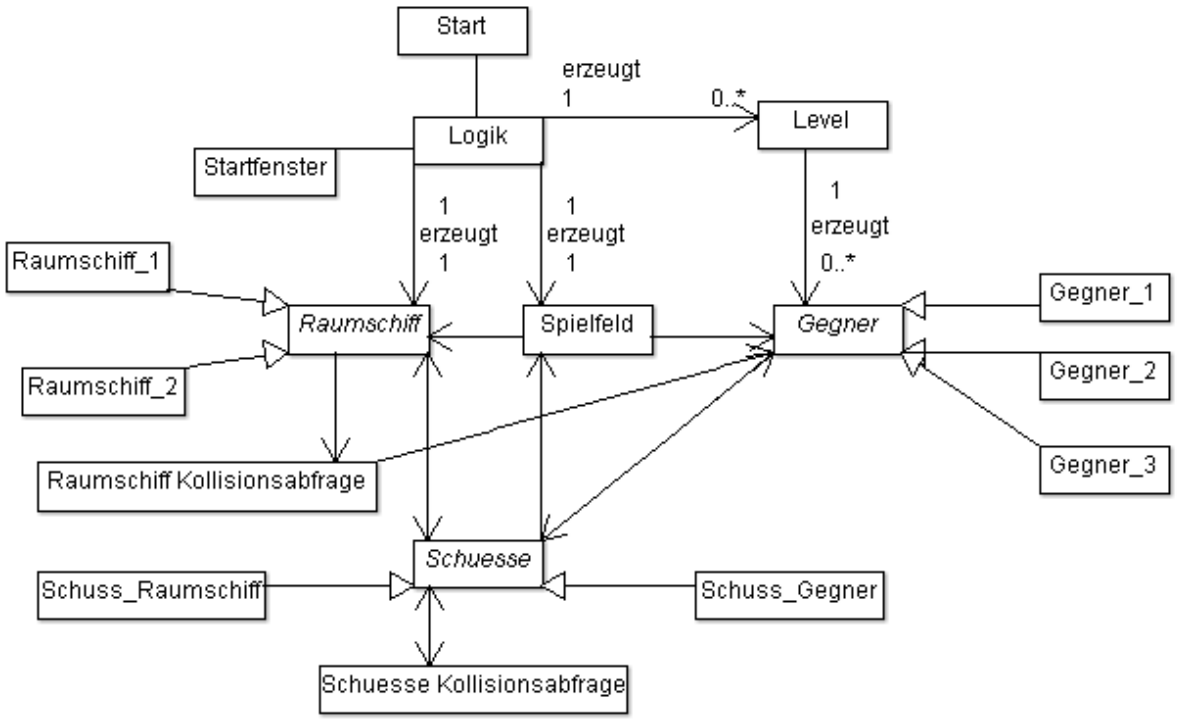
2. Zeit

Die Klassenstruktur hatten wir nach zweimal Ct fertig jedoch haben wir ständig neue Klassen programmiert die zuerst nicht eingeplant waren.
In den Herbstferien hatten wir eine erste Version, bei welcher man schon das Raumschiff steuern konnte. Doch wir stoßen auf das Problem, dass wenn man Gegner eingefügt hat das Spiel geruckelt hat und nicht flüssig lief. Uns wurde schnell klar, dass wir eine übersichtliche und möglichst einfache Struktur brauchen, die vor allem schnell funktioniert und nicht durch lange Rechnungen aufgehalten wird. Da das erste Programm sehr viel System Ressourcen

gebraucht hat, haben wir durch längeres überlegen, recherchieren und ausprobieren herausgefunden, dass vor allem unsere Oberfläche nicht optimal Programmiert ist. Der Fehler bestand darinnen, dass wir jedes einzelne Objekt als ein einzelnes Panel auf die Oberfläche gezeichnet haben und dieses für jede einzelne Position neu. Also haben wir nach einfacheren und weniger Rechenintensiven Methoden gesucht. Letztlich haben wir uns dazu entschieden die Graphics-Methode in Java zu überschreiben und mit der repaint()-Methode die einzelnen Elemente auf die Oberfläche zu zeichnen. Der Vorteil bestand nicht nur darinnen, dass diese Methode deutlich Ressourcen sparender ist, sondern auch wesentlich leichter zu handhaben. Die Methode wird nämlich von selbst während der Laufzeit ständig aufgerufen, weswegen wir nur die Positionen der einzelnen Elemente verändern mussten und die Grafik stellt dieses dann automatisch Bildlich da. Da wir nun das Problem mit den Ressourcen sparenden Programmieren gelöst hatten, machten wir uns nun Gedanken darüber, wie wir einen möglichst flüssigen, Ablauf im Programm, ermöglichen können. Uns war klar, dass das Programm zu komplex werden würde und somit ein einfach ablaufendes Programm keinen flüssigen Ablauf garantieren könne. Deswegen haben wir das Programm „Multi-Threading“ basiert programmiert. Durch die Hintergrund Anwendungen war es möglich auch große Rechenoperationen auszuführen, ohne dass das Programm ins Stocken gerät. Man kann sagen, dass wir mit diesen Überlegungen den Grundstein für unser Programm gelegt haben, in denen wir bis zum jetzigen Zeitpunkt keinen großen Fehler sehen, da sie auch das Programmieren deutlich vereinfacht haben. Die neue Struktur und vor allem auch die vielen neuen Funktionen, die wir noch zusätzlich hinzugefügt haben, sind der Grund warum sich die Struktur in unserem Programm sich Großteiles verändert hat.

3. Aufbau

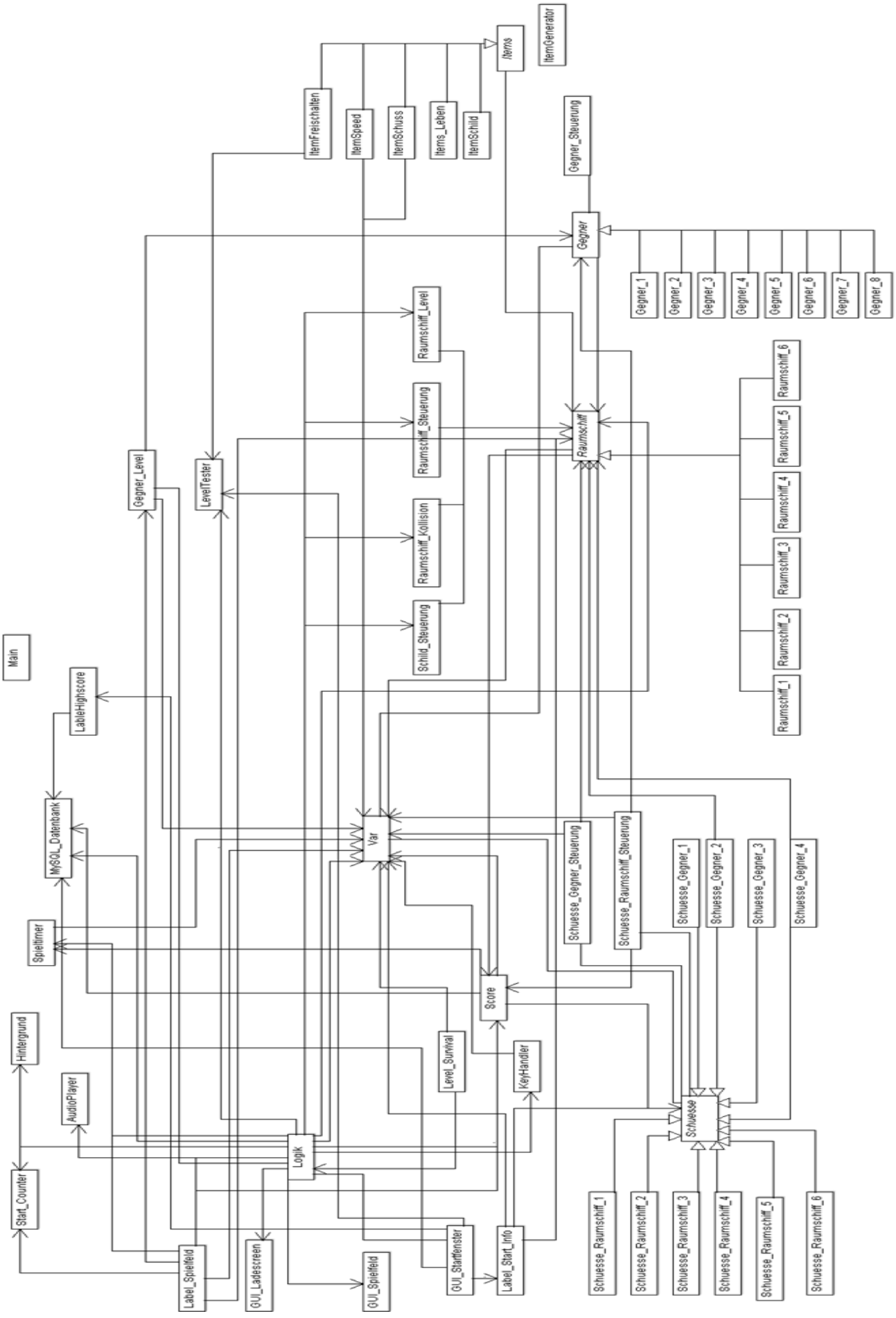
Unser erster Aufbau sieht so aus:



Das Spielfeld fragt das Raumschiff und die Gegner nach ihren Positionen und zeichnet sie dann. Die Gegner werden als Array in Level gespeichert und für jedes Level werden unterschiedliche Gegner erzeugt. Die Objekte vom Raumschiff und von den Gegnern sind jeweils Unterklassen den abstrakten vererbenden Klassen Raumschiff und Gegner. Durch Polymorphie wird während der Laufzeit entschieden welche Methode vom Raumschiffobjekt oder von den Gegnern aufgerufen wird. So kann man mit wenig Programmcode schnell neue Gegner und Raumschiffe erstellen.

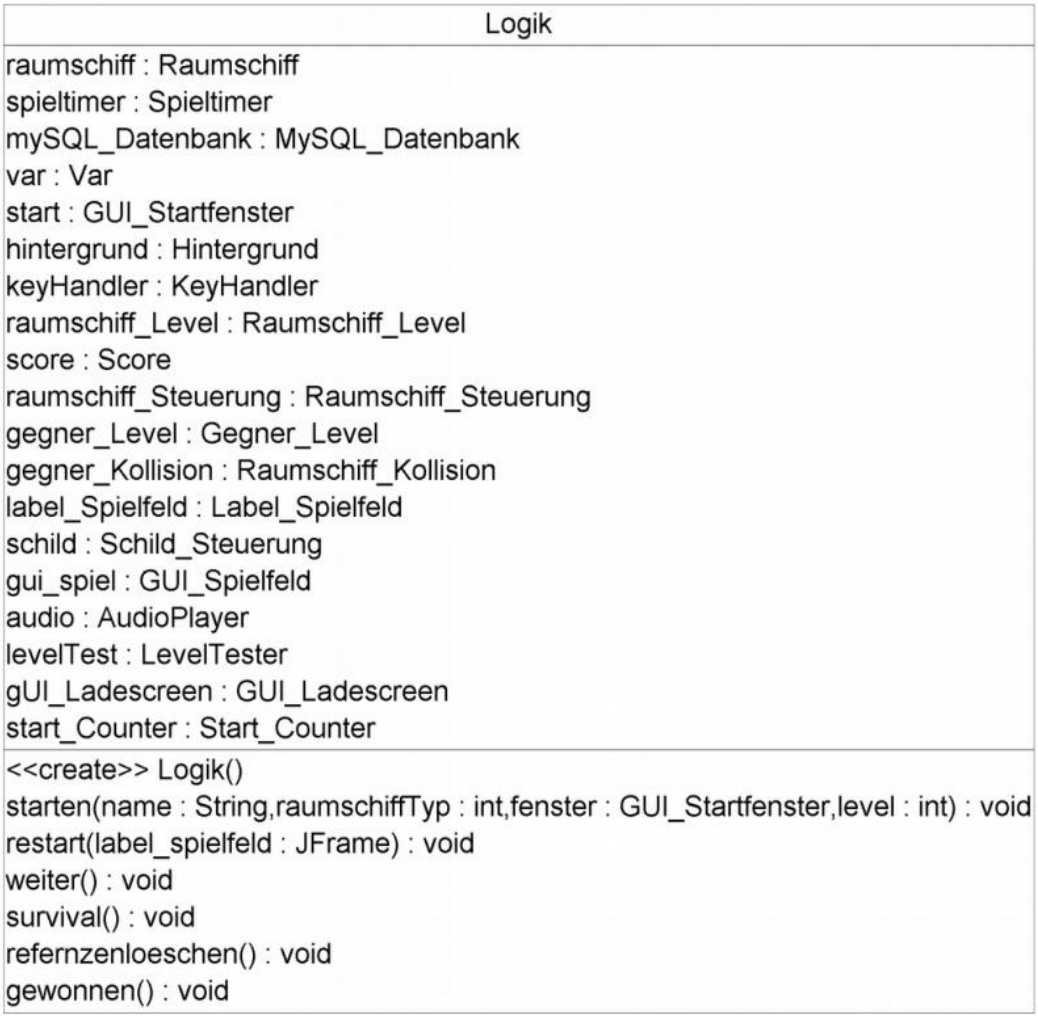
Jedoch mussten wir einiges an unserem Aufbau ändern.
Das fertige Klassendiagramm sieht so aus:

Computer Technik
Dokumentation Halbjahr 1



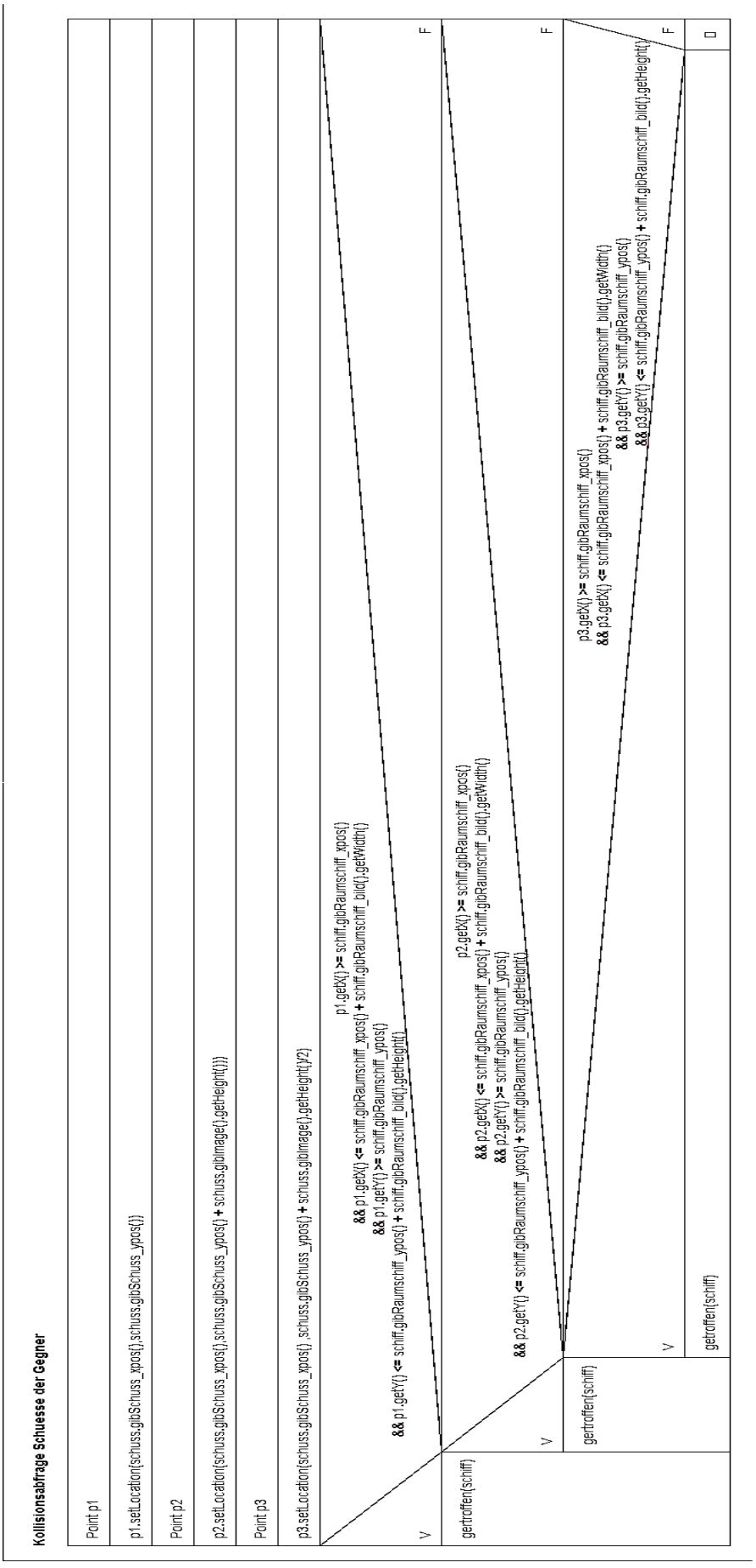
Torben, Nils

In dem fertigen Klassendiagramm sieht man, dass sich die Grundstruktur nur in einem Punkt verändert hat. Und zwar werden ein paar Variablen, wie die Bildschirmgröße oder welches Lied gerade gespielt wird in der Klasse Var gespeichert. Auch wird das Gegner Array und die Arrayliste mit Schüssen in der Var Klasse gespeichert. Das Gegner Array wird darin gespeichert, damit man die Gegner einfach verändern kann ohne dass man die neuen Gegner überall durchreichen muss. Bei den Schüssen ist das genau das gleiche. Immer wenn ein Schuss dazu kommt müsste das Schuss Array neu durchgegeben werden. Sonst hat sich nicht wirklich was verändert es sind noch viele neue Klassen, wie die Items oder neue Gegner und Raumschiffe dazu gekommen. Außerdem wurden Klassen wir AudioPlayer und LevelTester hinzugefügt um mehr Funktionen zu haben. Durch die oben genannten Klassen ist es möglich Musik zu spielen und das man sich die Level freispielen muss. Wir haben nur die Klassennamen im Klassendiagramm gezeigt, weil einige Klassen sehr viele Attribute haben und man es sonst nicht lesen konnte. Deswegen stelle ich jetzt die Klasse Logik vor.

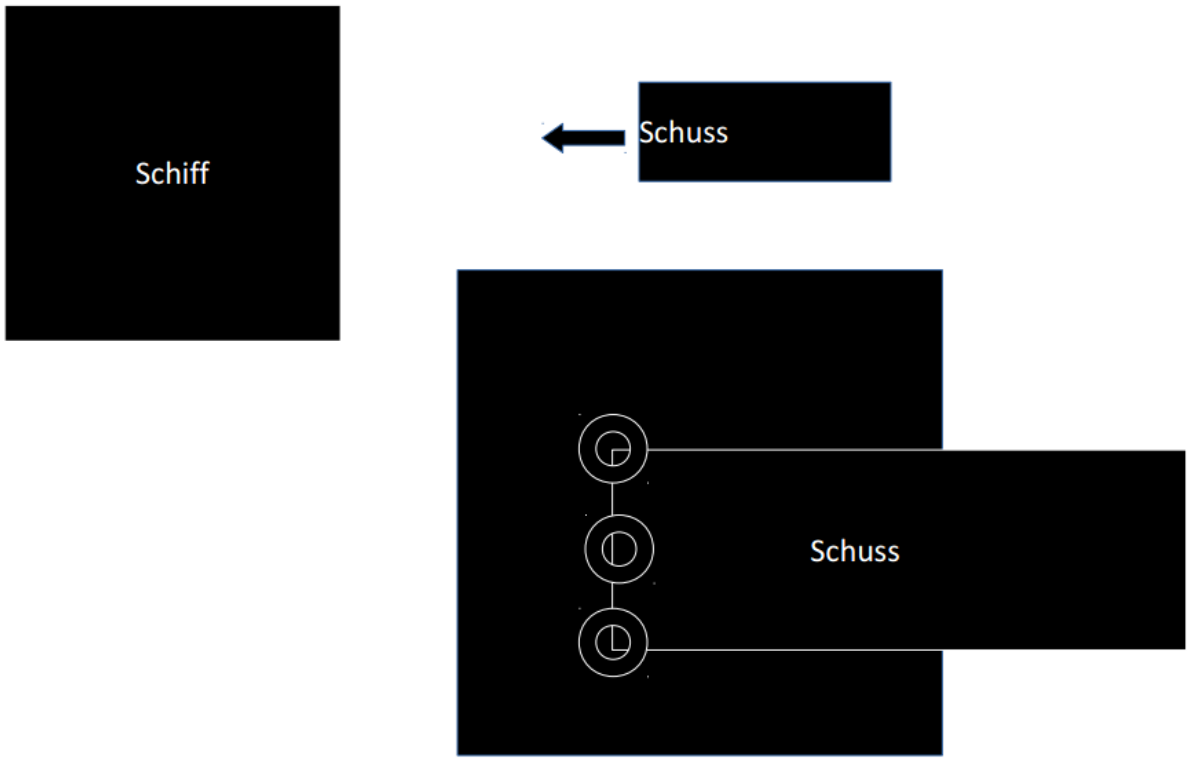


Die Klasse Logik hat fast alle Objekte als Attribute. Diese kann sie löschen und dann kann das Spiel neu gestartet werden. Das Spiel wird neu gestartet, wenn die Methode restart() aufgerufen wird. Hier wird dann ein neues Startfenster erstellt und die Referenzen werden gelöscht. Mit der Methode weiter() kann man nach einem erfolgreichen Level ein Level weiter springen ohne ins Startfenster zu gehen. Die Methode survival() ist für den Survival Modus zuständig. Wenn sie aufgerufen wird, dann ändern sich die Gegner aber der Rest bleibt gleich. Mit starten() wird nach dem Startfenster das Spiel mit den gewünschten Einstellungen gestartet.

Computer Technik
Dokumentation Halbjahr 1



Das Struktogramm zeigt die Kollisionsabfrage der Schüsse von den Gegnern. Zuerst werden 3 Punkte erstellt und dann wird gefragt ob einer dieser Punkte innerhalb vom Bild des Schiffes ist.



Bei diesem Bild sieht man die drei Punkte umkreist. Wenn der Schuss jetzt auf das Schiff zufliegt wird getestet, ob der Schuss innerhalb des Quadrates des Raumschiffes liegt. Ist dies der Fall dann wird die Methode getroffen aufgerufen und das Raumschiff bekommt Schaden.

Das Programm ist sehr auf Polymorphie ausgelebt und deswegen sind die einzelnen Klassen der Raumschiffe, Gegner und Schüssen nicht so groß. Es gab Probleme bei der Grafik und beim Neustart des Spiels, da durch static-Variablen es zu Fehlern kam. Jedoch wurden die Probleme behoben und das Spiel funktioniert. Wir können nicht ausschließen, dass es keine Bugs mehr hat aber um das zu testen dauert es ziemlich lange.

4. Anleitung

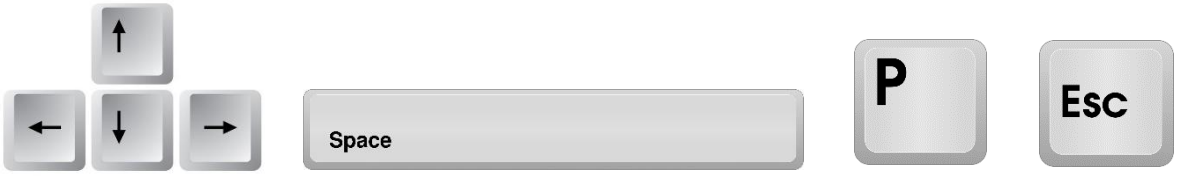
SPACE INVADER RELOADED

by Nils und Torben

Space Invader beziehungsweise hier Space Shooter gehört zu eines der ersten Videospiele und wurde im Jahre 1978 das erste Mal veröffentlicht. Es fällt unter die Kategorie „Shoot-'em-up-Computerspiel" und es gibt bis heute hunderttausende Remakes vom damaligen Spieleklassiger. Das Ziel des Spiels besteht nicht nur darinnen zu überleben, sondern auch darinnen einen möglichst hohen Score zu erreichen um in der Rangliste möglichst weit nach oben zu kommen.

- **Steuerung:**

Im Spiel steuern Sie ein Raumschiff, dass Sie zu Beginn des Spiels auswählen können. Man bewegt das Raumschiff mit den Pfeiltasten (hoch, runter, rechts, links) und schießen mit der Leertaste. Sie können das Raumschiff auf dem kompletten Bildschirm bewegen, aber versuchen Sie dabei anderen Schiffen auszuweichen und nicht zu kollidieren. Falls Sie währen dem Spiel eine Pause machen wollen benützen Sie einfach die Pause Taste („P"). Das Programm beenden, während dem Spiel, kann man mit der Esc-Taste.



- **Level:**

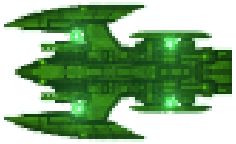
Wie in vielen Spielen gibt es auch hier Level die Sie erst freischalten müssen, um sie spielen zu können. Auf dem Startscreen kann man sehen wie viele Level Sie bis jetzt freigespielt haben. Um ein weiteres Level freizuschalten müssen Sie das Level davor erfolgreich absolvieren (Ein Beispiel: Sie wollen Level 6 spielen, also müssen Sie um Level 6 freizuschalten Level 5 erfolgreich absolvieren). Man hat ein Level erfolgreich geschafft, wenn man die vorgegebene Zeit, die je nach Level variieren kann, überstehst ohne deine Leben zu verlieren. (Wichtig: Das Schild ist nur ein Bonus, man besteht das Level auch wenn man sein Schild vollständig verloren hat!)

- **Raumschiffe:**

Im Spiel hat man mehrere verschiedene Raumschiffe zur Auswahl. Jedes Raumschiff hat seine eigenen Stärken und Schwächen, die Sie im Startmenü entnehmen können. Finden Sie heraus welches Raumschiff am besten zu ihnen passt und welches für die einzelnen Level am besten ist. Manchmal ist die Geschwindigkeit der entscheidende Faktor, aber auch der Schaden kann in Leveln für das Überleben von großer Bedeutung sein. Wie auch bei den Leveln, muss man die Raumschiffe erst freispielen. Sammeln Sie dazu einfach dieses grüne Icon in den Leveln ein, um ein weitere Raumschiffe freizuschalten.

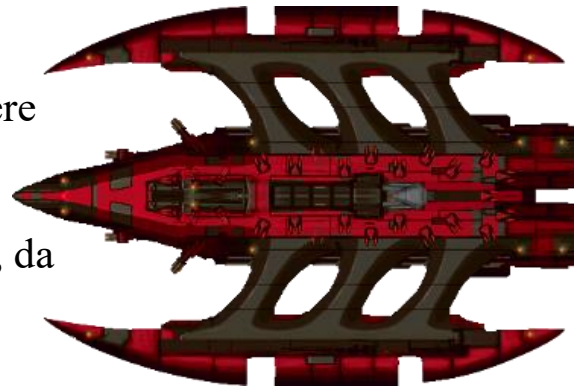
- **Schild:**

Jedes Raumschiff besitzt ein eigenes Schild. Das Schild absorbiert den erlittenen Schaden. Es regeneriert sich alle 8 Sekunden (Wichtig: nicht bei jedem Raumschiff regeneriert das Schild gleich schnell!).
Das Schild befindet sich nur vorne, alles was dich außerhalb der Reichweite des Schildes trifft beschädigt direkt die Leben deines Raumschiffes und wir nicht durch das Schild abgefangen.



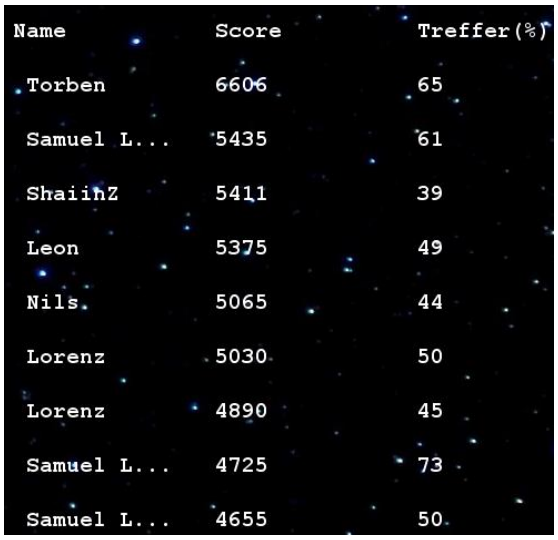
- **Gegner:**

Wie auch bei den Raumschiffen, haben auch die Gegner Stärken und Schwächen. Finden Sie heraus, vor welchen Sie sich besser in Acht nehmen und wie man sie am besten besiegt. Manche können Sie mit ein bis zwei treffen vernichten, für andere brauchen Sie deutlich länger.
Sorgen Sie dafür, dass die Anzahl der Raumschiffe nicht zu groß wird, da sonst das Ausweichen schier unmöglich wird. Wenn ein Gegner durchkommt, wird es nicht negativ angerechnet. Das Ziel des Spiels besteht nicht darinnen, keine Gegner durchzulassen, sondern die höchst mögliche Anzahl an Gegnern zu vernichten.



- **Score:**

Ihre Punkte sind mit das wichtigste im Spiel, denn mit ihnen und ihrem Nickname, den Sie zu Beginn des Spiels bestimmt haben,



Name	Score	Treffer(%)
Torben	6606	65
Samuel L...	5435	61
ShaiihZ	5411	39
Leon	5375	49
Nils.	5065	44
Lorenz	5030	50
Lorenz	4890	45
Samuel L...	4725	73
Samuel L...	4655	50

tragen Sie sich später in die Highscoreliste ein. (Ein Eintrag erfolgt nur, wenn du das Level schaffst!) Hauptsächlich sammelt man Punkte durch das Vernichten der Gegner, aber es gilt dabei auch drauf zu achten, dass man nicht zu oft danebenschießt, denn je besser die Treffergenauigkeit ist, umso mehr Punkte werden dem Spieler zusätzlich noch gutgeschrieben (Der aktueller Score, denn man durch vernichtete gegnerische Einheiten erhalten hat, wird plus den aktueller Score Mal der Treffergenauigkeit gerechnet. Bei einer Treffergenauigkeit von 50% hätte man dann also das 1,5-fache seines Scores). Punkte werden Ihnen abgezogen, wenn Sie mit einem Gegner kollidieren, oder Sie ein Schuss trifft. Am Ende jedes erfolgreichen Spiels wird Ihnen der erreichte Platz angezeigt. Versuchen Sie unter die Top 10 zu kommen um auch für die anderen Spieler sichtbar zu sein und werden Sie einer der besten Space-Piloten im Universum.

- **Bonus-Items:**

Die Bonus Items begegnen Ihnen in jedem Level unterschiedlich oft. Sammeln Sie sie ein um kurzfristig stärker zu werden oder um ihr Raumschiff oder Schild zu reparieren.

Dieses Item ermöglicht ihnen, dass Sie sich für eine kurze Zeit besonders schnell bewegen können, achte dabei aber darauf, dass du trotzdem noch den Gegnern gekonnt ausweichst.

Dieses Item sorgt für eine deutlich höhere Schussrate. Nutzen Sie es um sich wieder ein bisschen Platz zu verschaffen.

Mit diesem Item können Sie ihr Schild wieder ein Stück aufladen.

Das Herz ist die einzige Möglichkeit verlorene Leben wieder zu gewinnen, und so länger im Weltall zu überleben.

Mit diesem Item schalten Sie ein weiteres neues Raumschiff frei.

5. Zusammenfassung

Das Projekt ist gut gelungen. Nachdem wir die Probleme mit der Grafik behoben hatten konnten wir ständig neue Sachen hinzufügen. Wir haben herausgefunden, dass das Debuggen viel schwerer ist als das eigentliche programmieren. Da haben uns unsere Klassenkameraden sehr geholfen, weil diese das Spiel gespielt haben und uns auf Fehler hingewiesen haben. Außerdem haben diese uns auch Verbesserungsvorschläge gemacht und diese haben wir dann meistens auch umgesetzt. Jetzt kann man auch noch weiter neue Level, Gegner oder Raumschiffe hinzufügen und das Programm erweitern. Wir hatten eine gute Teamarbeit, da wir früher auch schon zusammen an Projekten programmiert haben und wir beide ganz gut programmieren können. So mussten wir nicht dem Anderen helfen und konnten eigenständig das Programm erweitern. Durch Git und Bitbucket war eine bessere Teamarbeit möglich. Bitbucket ist eine Website welche auf Git basiert. Man kann mit dieser zusammen als Team arbeiten indem man immer seine eigenen Veränderungen durch ein Programm hoch lädt. Die Änderungen des Teams werden automatisch zusammengefügt und man kann auch einfach auf frühere Versionen zurückgreifen. Das heißt, dass wenn man mal etwas programmiert hat das dann doch nicht so toll war kann man diese Änderungen leicht rückgängig machen. Mit diesem Programm konnten wir beide Zuhause programmieren ohne ständig von Hand die Änderungen zusammenzufügen. Die Teamarbeit ist also sehr gut gelungen. Wir hatten Spaß an dem Projekt und haben es auch zum Teil als Hobby weiter programmiert.

6. Quellen

Wissensquellen:

- Java 8 Api (https://docs.oracle.com/javase/8/docs/api/)
- https://www.dpunkt.de/java/Referenz/Das_Paket_java.awt/59.html
- http://openbook.rheinwerk-verlag.de/javainsel9/javainsel_20_001.htm

Bilderquellen:

- http://opengameart.org/content/space-ship-construction-kit
- http://vignette1.wikia.nocookie.net/pixelstarships/images/c/cf/Shield.png/revision/latest?cb=20151126045036
- http://voidwar.deviantart.com/gallery/
- https://cdn4.iconfinder.com/data/icons/proglyphs-free/512/Invader_1-512.png