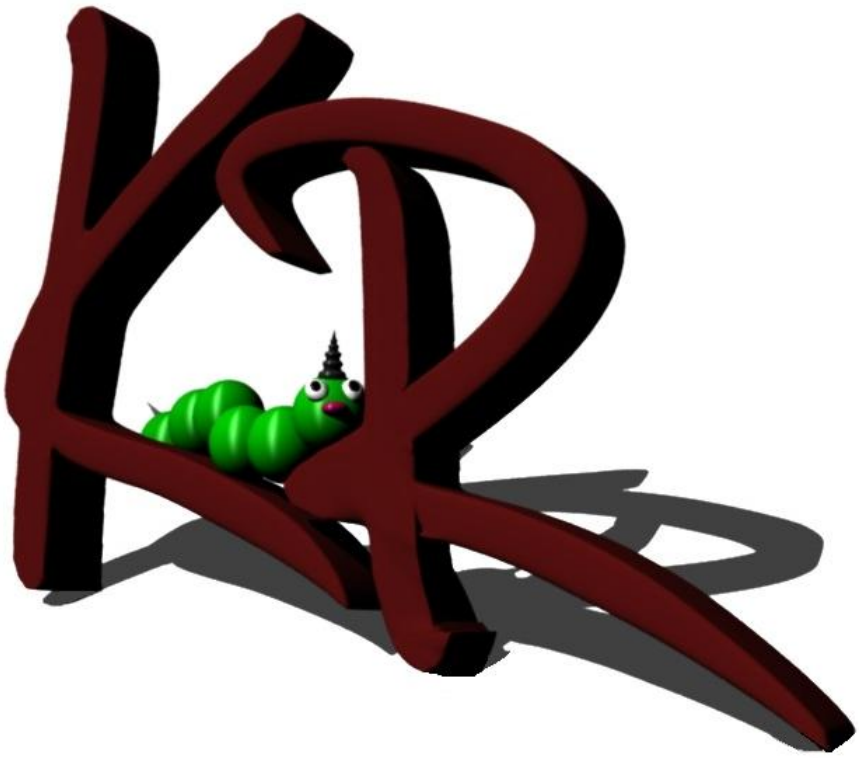
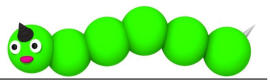
Dokumentation des Projektes Spaceballs

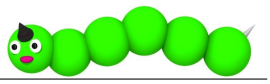
der Gruppe KillerRaupi

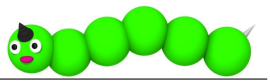
29. Januar 2015





Inhaltsverzeichnis





1 Projektvorfeld

1.1 Aufgabenstellung

Das Projekt „Spaceballs" ist ein Lehrprojekt im Modul Informatik des Studiengangs Luftfahrtsystemtechnik und -management (ILST) der Hochschule Bremen. Das Projekt wird von den ca. 60 Studierenden des ILST durchlaufen, die sich dazu in neun Gruppen mit bis zu sieben Mitgliedern zusammen finden. Die Aufgabe des Projektes „Spaceballs" besteht darin, eine Künstliche Intelligenz (KI) für ein Spiel zu programmieren, die -optimaler Weise- eine andere KI besiegt und gewinnt[1]. Die Programmierung erfolgt mit dem Programm MATLAB, wobei ein einziges File (mit verschiedenen Unterprogrammen) geschrieben werden muss, welches die Bewegung des Spaceballs steuern soll.
Des Weiteren muss eine Website designed werden, auf welcher sich die Projektgruppe und ihr zugehöriger Spaceball ansprechend vorstellt und Verlinkungen vorhanden sind, welche auf die Dokumentation zu genaueren Informationen verweisen. Das ist direkt die nächste Aufgabe: Es muss eine Dokumentation über das gesamte Projekt erstellt werden, wo der Projektfortschritt festgehalten und genaue Erklärungen abgegeben werden.
Darüber hinaus sollen sechs Videos erstellt werden, die dem Spielgeschehen mehr Wirkung und einen höheren Spaßfaktor verleihen. Diese sollen in 3-D animiert werden und dürfen keine kopierten Inhalte enthalten, sollen also zu 100% von den Studierenden erstellt worden sein. Außerdem muss der Spaceball in eine andere „Figur" o.ä. umgewandelt werden, die ebenfalls animiert werden soll und auch gewisse „Interaktionen" ausführen können sollte.
Darüber hinaus sind die Studierenden verpflichtet jede Woche in ein Online-Projekttagebuch einen Eintrag zu erstellen, wo sie beschreiben mit welchen Inhalten sie sich in dieser Kalenderwoche beschäftigt haben und welche persönlichen Fortschritte sie in ihrem Teilgebiet (Programmierung, Websitedesign, Animation, Dokumentation) erzielt haben. Im Übrigen muss jedes Gruppenmitglied über das gesamte Semester verteilt fünf Minipräsentationen vor dem Professor halten, wo der Stand des Projektes erläutert, Fragen beantwortet und Fragen gestellt werden sollen. Diese Maßnahme dient dazu sicherzustellen, dass sich jedes Mitglied an der Arbeit am Projekt regelmäßig beteiligt. Insgesamt ergibt sich eine Erarbeitungszeit von 13 Semesterwochen. Vorlesungen werden grundsätzlich keine angesetzt, es sei denn es besteht eine erhöhte Nachfrage.
Zwei Wochen vor Ende des Semesters müssen die KIs programmiert, sowie das gesamte „Layout" (Videos, Animation...), die Website und die Dokumentation fertig gestellt worden sein, denn dann messen sich alle Spaceballs in einem „jeder-gegen-jeden-Tunier". Wer gewinnt erhält einen bis zwei Punkte, sodass eine Rangliste entsteht, die zu 50% in die Bewertung eingeht. Die restlichen 50% ergeben sich aus der Bewertung der Dokumentation, den Videos der Animation und der Website. Letztere wird nach dem Turnier online-geschaltet.

[1] siehe Kapitel 1.2 „Das Spiel Spaceballs und seine Regeln"

1.2 Das Spiel Spaceballs und seine Regeln

Bei den Spaceballs handelt es sich um gefräßige Alien-Kreise, deren Lebensziel darin besteht so viel grüne Tankstellen wie möglich leerzusaugen um maximal viel Treibstoff zu sammeln, schwarzen Löchern - auch Minen genannt - auszuweichen und nicht die Begrenzung des Spielfeldes zu berühren. Es gewinnt der Spaceball den Battle, der mehr Sprit hat und den Gegner in diesem Zustand berührt. Unterm Strich also Fangen-Spielen für Schwergewichtskreise mit erhöhtem Schwierigkeitsgrad.
Wenn ein Spaceball „tankt", dann entzieht er einer Tankstelle Treibstoff - deren Radius schrumpft - und wird dabei selbst größer. Wenn sich ein Spacecball allerdings bewegen möchte, dann verbraucht er natürlich Sprit und wird kleiner, indem er mit diesem eine Art raketenähnlichen Antrieb betreibt, der ihn mittels Schub, also des Rückstoßprinzips, in die gewünschte Richtung bringt. Hierbei bewegt sich der Spaceball angenehmerweise im Vakuum, das heißt da keine Reibung auftritt, wird der Spaceball nie langsamer, sondern behält seine Geschwindigkeit auch bei, wenn der Schub abgestellt wird und gleitet somit weiter.
Die Ausgangssituation in der Spaceball-Arena sind in der nächsten Abbildung zu sehen.

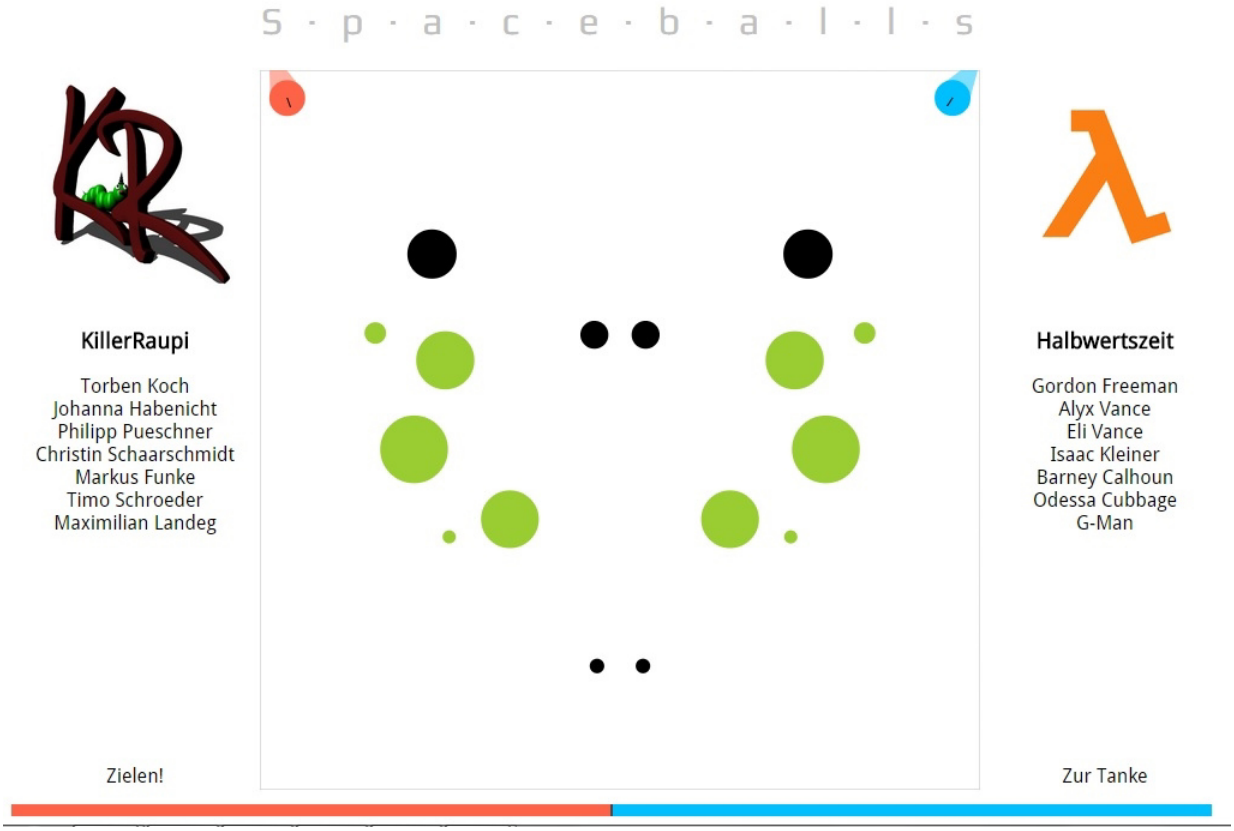


Abbildung 1: Ausgangssituation des Spieles

Die Spaceball-Arena besteht aus einer Bande, die das Vakuum begrenzt. Wenn ein Spaceball sie berührt, hat er verloren. Weiterhin befinden sich in der Arena sechs schwarze Minen, die unterschiedliche Größen haben können und deren Verteilung variiert. Des Weiteren gibt es in der Arena zehn Tankstellen mit Sprit, schließlich bietet

der Treibstoff die Grundlage des Spiels. Die Lage und Größe der Minen und Tankstellen werden für jedes Spiel per Zufall festgelegt, das heißt die Arena sieht in jedem Spiel anders aus, so bleibt es spannend und es ist auch unmöglich einfach seinen Spaceball eine optimale Route mit spezifischen Wegpunkten einzubläuen. Die Anzahl der Tankstellen und Minen ist jedoch immer konstant, ebenso die Größe der Arena. Die Minen und Tankstellen werden symmetrisch angeordnet, damit keinem Spaceball eine Benachteiligung entsteht, denn das Spiel beginnt für den roten Spaceball in der oberen linken und für den blauen Spaceball in der oberen rechten Ecke.
Dann haben die Spaceballs 120 Sekunden lang Zeit Sprit zu tanken und zu wachsen und den kleineren Gegner (im Optimalfall) zu fangen und zu berühren.
Der erste Weg führt natürlich immer zuerst zur Tankstelle, wie auch in der nächsten Abbildung zu sehen ist.

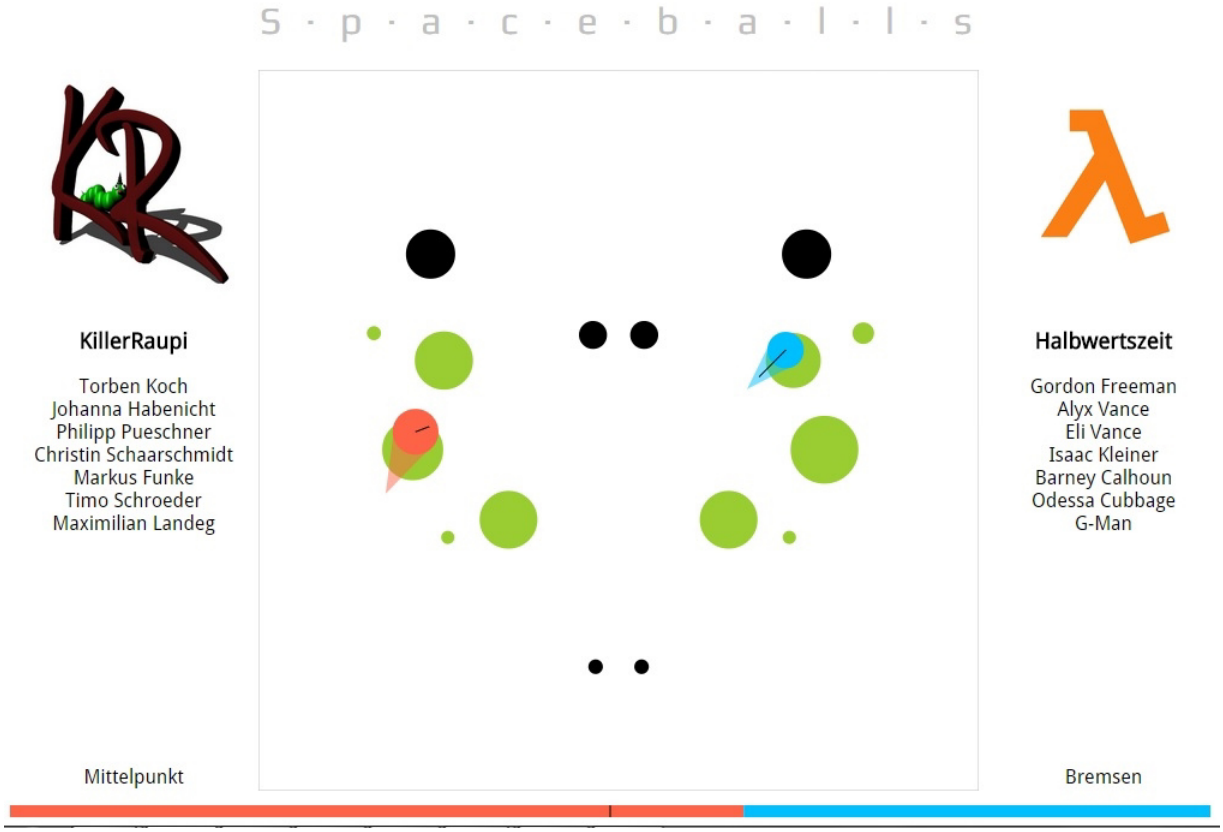


Abbildung 2: Tanken um zu Wachsen

Wenn ein Spaceball eine respektable Größe glaubt erreicht zu haben, dann muss er den Gegner berühren und das Spiel ist beendet. In diesem Fall gewinnt der Spaceball, der größer ist und erhält in der Gesamtwertung 2 Punkte. In Abbildung 3 ist ein solcher Endstand zu sehen, in diesem Fall hätte also der rote Spaceball 2 Punkte erhalten. In jedem Fall endet das Spiel, wenn sich die Spaceballs berühren, auch wenn der Kleinere einen Fehler begeht und in den Größeren hineinfährt, dann hat der Kleine Pech und verliert.
Wenn nach 120 Sekunden es keinem Spaceball gelungen ist den anderen zu berühren oder das vielleicht auch gar nicht gewollt worden ist, dann endet das Spiel

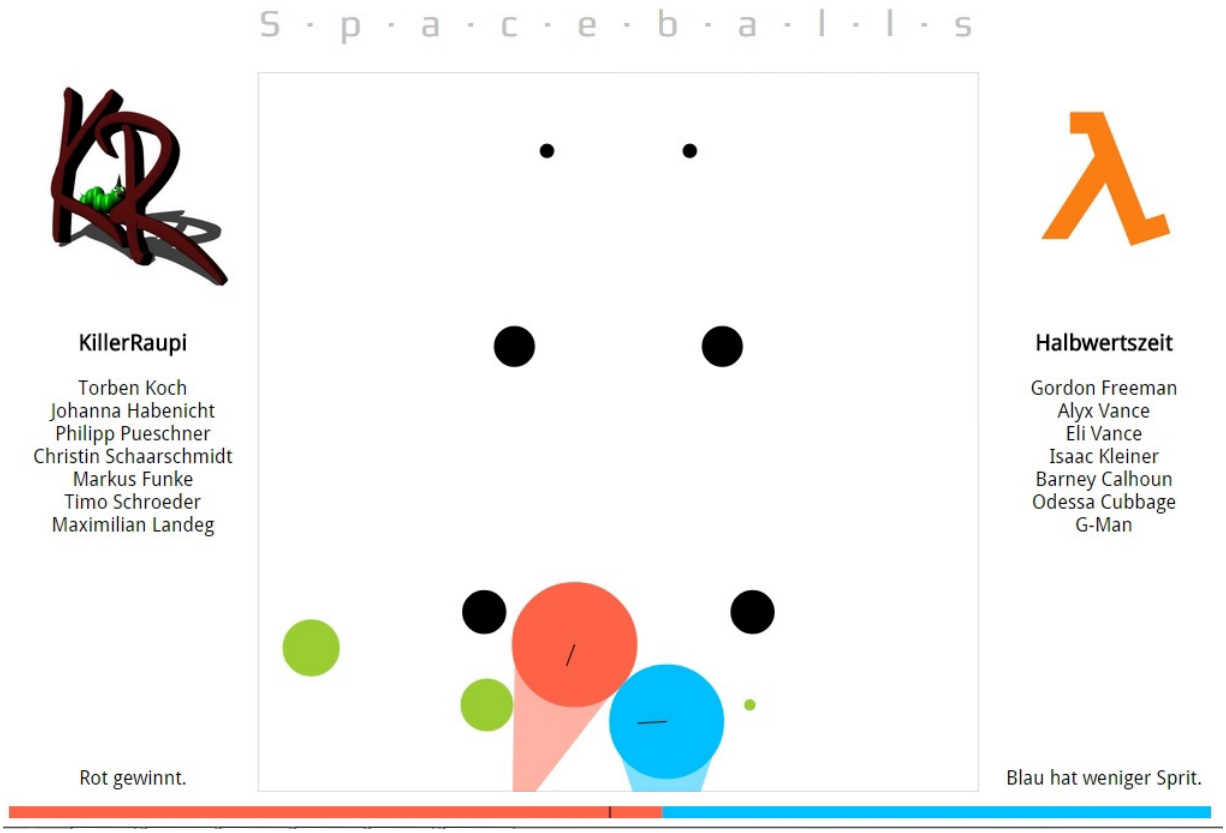
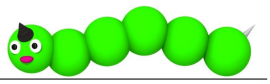


Abbildung 3: Spielende: Rot gewinnt

mit einer Art Unentschieden, wobei allerdings der Spaceball in der Gesamtwertung einen Punkt erhält, der mehr Sprit ertankt hat und somit größer ist.
Die Regelung, dass der Spaceball Punkte erhält, der größer ist, tritt dann außer Kraft, wenn ein Spaceball eine Mine oder eine Bande berührt, dann erhält der übrigbleibende Spaceball 2 Punkte. Das suggeriert, dass es einen Fehler in der Programmierung der KI gibt und dementsprechend darf man in der Gesamtwertung keine Punkte erhalten.

Welche Gruppe mit ihrer KI am Ende des Turniers in der Gesamtwertung die größte Punktzahl erreicht hat, wird zum offiziellen Sieger gekürt.



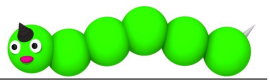
1.3 Zielsetzung

Die übergeordnete Zielsetzung besteht natürlich darin alle gestellten Aufgaben zu erfüllen. Dabei stellen wir allerdings noch besondere Anforderungen an uns selbst und an das Resultat, denn wir möchten nicht nur die Aufgaben erfüllen, sondern wir möchten sie gut erfüllen.
Für die Animation haben wir uns grundsätzlich vorgenommen eine comichafte Raupe in 3D zu animieren, die dann auch umher kriecht. Sie soll in allererster Linie grün sein, ein niedliches Erscheinungsbild besitzen, aber auch sympathisch und witzig sein. Trotzdem wird sie große scharfe Zähne haben, die einen Kontrast zu der klischeehaften Niedlichkeit stehen sollen. Anstatt wie der „normale" Spaceball zu tanken, soll KillerRaupi Blätter fressen und dann natürlich auch an Volumen zunehmen. Als weitere Videosequenzen sind geplant die Raupe explodieren zu lassen, wenn sie an eine Mine fährt. Wenn KillerRaupi trotz bester Programmierung in eine Wand fährt, dann soll das Video zeigen wie sich Raupi an der Wand zusammenschiebt und nur ein Fleck bleiben. Wenn die Raupe gegen eine andere KI verliert, dann wird sie herzzerreißend weinen und traurig sein. Für den Sieg hingegen gibt es sogar zwei Enden: Zum einen, wenn Raupi gewinnt weil sie nur mehr Sprit hat als der andere, dann soll sie einfach im Video als gigantisch groß und dick erscheinen. Wenn sie gewinnt, weil sie den anderen Spaceball berührt und mehr Sprit hat, dann wird sie natürlich zu einem wunderschönen Schmetterling.

Als Inhalt für die Website ist zum ersten eine Home-Seite angedacht, wo der Inhalt des Spieles „Spaceballs" bereits kurz erklärt wird und Raupi kurz vorgestellt wird.
Zum anderen soll eine Seite existieren, die das Spiel „Spaceballs" im Detail erläutert und erklärt und alle Regeln beinhaltet.
Außerdem wird es natürlich ein Impressum und einen Kontakt-Link geben, denn KillerRaupi besitzt eine eigene Email-Adresse.
Des Weiteren wird auf der Website auch das Projektteam vorgestellt. Dabei ist zu jedem Mitglied ein Foto und Beschreibung seiner Aufgaben im Projekt vorhanden.
Der Kerninhalt der Website wird natürlich KillerRaupi selbst sein. Eine individuell angepasste Hintergrundgeschichte, ihre Entwicklung und natürlich in allererster Linie der Zusammenhang zu dem Spiel Spaceballs.

Die Programmierung der KI steht bei allen Aufgaben natürlich weitestgehend im Mittelpunkt und hat auch die meisten Mitarbeiter. Die KI soll an erster Stelle zuverlässig tanken können, nicht gegen Wände und Minen fahren, gezielt und intelligent angreifen können, aber sich auch gegen eventuelle Angriffe verteidigen können, ohne in die Bande oder gegen Minen gedrängt zu werden.
Als zeitliche Abfolge haben wir uns daher gesetzt:

1. Tanken
2. Bremsen
3. Wand
4. Angriff
5. Verteidigung
6. Minen



1.4 Angestrebte Strategie unserer KI

Wie bereits in Kapitel 1.3 beschrieben, soll die KI natürlich grundlegende Tätigkeiten durchführen können, wie: Tanken, Bremsen, Wände und Minen nicht treffen, sowie Angreifen und Verteidigen können.
Oberste Priorität hat unserer Meinung nach, dass die KI so programmiert ist, dass sie absolut zuverlässig keine vermeidbaren Fehler begeht, die uns den Sieg kosten würden, beispielsweise gegen die Wand oder in Minen zu fahren. Wir stellen also an uns den Anspruch nicht durch vermeidbare Fehler zu versagen.
Des Weiteren soll KillerRaupi nicht angreifen, wenn sie an einer Tankstelle ist, da wir grundsätzlich beschlossen haben, dass das Tanken hohe Priorität besitzt, da Sprit eine begrenzte Ressource ist, die allerdings entscheidend für das Gewinnen des Spiels ist. Daraus leitete sich auch die Überlegung ab, dass es denkbar wäre den Spaceball maximal viel Sprit aufsaugen zu lassen und dann stehen zu bleiben. Also eine sehr passive Strategie zu verfolgen. Dies ist allerdings nur eine Überlegung gewesen, es ist aber recht unwahrscheinlich, dass wir uns darauf einigen diese Strategie zu verfolgen. Denn wir haben beobachtet, dass KillerRaupi schon beim Tanken in manchen Situationen dem Gegner sehr nahe gekommen ist und problemlos hätten gewinnen können, wenn KillerRaupi schon in der Lage gewesen wäre anzugreifen.
Allerdings soll der Gegner dann nicht angegriffen werden, wenn er an der Tanke steht und durch das Tanken größer werden würde als KillerRaupi oder schon so nah an einer Tankstelle ist, dass er diese schneller erreichen würde, als KillerRaupi den Gegner. Damit soll auch vermieden werden, dass der Spaceball das Tanken abbricht um den Gegner anzugreifen, auf dem halben Weg merkt, dass das Angreifen doch keine gute Idee wäre und wieder umdreht. So besteht eine erhöhte Gefahr, dass sich der Spaceball festfährt und hängen bleibt.
Das sind soweit die ersten Gedanken, die uns so am Anfang in den Kopf schießen, vielleicht wird sich manches als überbewertet entpuppen und mit Sicherheit werden wir noch viel hinzulernen müssen, da wir bestimmt einige Gefahren in unseren Vorüberlegungen übersehen haben.

2 Das Team KillerRaupi

Abbildung 4: Die Teammitglieder

Das Team KillerRaupi besteht aus sieben Personen, alle studieren im ILST und sind im 2. Semesterverband angemeldet. Vor diesem Projekt hatten die einzelnen Mitglieder zum Teil sehr wenig miteinander zu tun, umso schöner war die Erfahrung, dass eine gemeinsame Aufgabe alle Beteiligten doch sehr eng zusammen schweißen kann und auch in der Freizeit viel miteinander unternommen wird.
Dieses Projekt war an sich einfach eine tolle Erfahrung, weil wir als Team wunderbar zusammengearbeitet haben und nicht zuletzt deswegen, weil jeder einzelne ein ganzes Stück Herzblut mit in dieses Projekt hineingesteckt hat.
Die praktische Anzahl unserer Teammitglieder ermöglicht es uns außerdem jedem Teilnehmer immer nur eine Aufgabe aus einem Bereich zu übertragen. Die Programmierung mit Matlab ist hierbei am stärksten vertreten, drei Leute werden eingesetzt um unsere KI mit einer erfolgbringenden Strategie auszustatten. Einer aus der Gruppe wird sich mit der Animation unserer KillerRaupi in Maya beschäftigen. Weiterhin waren wir in der komfortablen Position die Aufgaben Dokumentation und Website an jeweils eine Person zu verteilen. Unser absoluter Vorteil besteht allerdings in unserem Libero-Joker, der mal hier, mal da, neben der Programmierung aber vor allem in der Animation mitarbeitet.

Christin Schaarschmidt

Christin besaß bereits einige Vorkenntnisse über das Textsatzsystem Latex (gesprochen: Latech), weshalb die Entscheidung nicht schwer fiel ihr als Aufgabe die Dokumentation zu übertragen. Ihr vorrangiges Ziel besteht hierbei darin möglichst genau die Fortschritte der Gruppe über die Zeit festzuhalten und die Doku nicht zu steif, sondern eher lebendig zu gestalten. Christins Genauigkeit wird das Projekt KillerRaupi für alle nachfolgenden ILSTler bestehen lassen.

Johanna Habenicht

Johannas Aufgabe im Projekt wird in der Erstellung der Website bestehen, denn ihre Stärken liegen in besonderer Kreativität und einem guten Auge für Details. Deshalb wird die Website bei ihr gut aufgehoben sein, denn Johanna recherchiert genauestens und sie wird auch nicht eine rechtliche Unklarheit ungeklärt lassen, sondern alle Hebel in Bewegung setzen, damit eine solide Website online geht.

Markus Funke

Markus wird sich mit einem sehr hohen Zeitaufwand und umfangreichen Mathematikkenntnissen in die Programmierung unserer KI stürzen. Das Mittelpunkt-Tankverfahren geht auf seine Idee und Umsetzung zurück und gemeinsam mit Max Landeg wird er KillerRaupi erfolgreich um alle Minen herum manövrieren. Außerdem wird er unseren PCs viel Rechenaufwand durch clevere Vereinfachungen im Code ersparen.

Maximilian Landeg

Maximilian, genannt Max, hatte bereits vor diesem Projekt Programmiererfahrung gesammelt und wird folglich die Programmierung unserer KI mitgestalten und wesentlich voran bringen. Das Antiwand-System, das KillerRaupi davon abhält in die Bande zu rasen wird hauptsächlich von ihm erstellt werden und gemeinsam mit Markus wird die optimale Route zu jeder Tankstelle entworfen werden.

Philipp Püschner

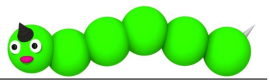
Philipp wird der Steven Spielberg unserer Gruppe KillerRaupi sein, denn er wird der ersten KillerRaupi der Welt ein Gesicht und einen Körper geben. Liebevoll wird er sie animieren und verbessern. Alle Videos starring KillerRaupi werden von Philipps Regiestuhl aus gedreht, geschnitten und gerendert werden.

Timo Schröder

Timo ist der Flexibelste von uns und ein Allround-Talent. Anfangs wird er ein wenig in der Programmierung mitmischen, recht bald allerdings wird er zur Animation überwechseln und KillerRaupi ein Zuhause geben, das Gruppenlogo entwerfen und KillerRaupi in den verschiedensten Posen animieren. Nebenbei steuert er auch immer gern einige Ideen zur Programmierung bei.

Torben Koch

Torben wird KillerRaupi auf Krawall bürsten, denn innerhalb der Programmierung wird er sich im speziellen mit dem Thema Angriff beschäftigen und unsere Raupe so nicht nur gefährlicher machen, sondern zu einer ernsthaften Bedrohung für alle anderen Spaceballs. Sollte KillerRaupi allerdings selbst einmal angegriffen werden, dann wird ihr Torben auch beibringen, wie sie sich zu verteidigen hat.



3 Dokumentation der Gruppentreffen

3.1 1. Meeting - 8.10.2014

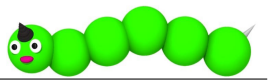
Das erste Gruppentreffen stand im Zeichnen der Organisation, Absprache und Diskussion. Unsere Gruppe hatte sich sehr schnell gefunden und formiert, da wir bereits vorher ein wenig spekuliert hatten, dass es sich in Informatik wahrscheinlich um ein Gruppenprojekt handeln würde.

Als erster Punkt der Tagesordnung kristallisierte sich dann eine lebhafte Diskussion über das Info-Projekt heraus, wo kritisiert und Vor- und Nachteile abgewogen wurden. Des Weiteren versuchten wir Prioritäten und Schwerpunkte zu setzen, die uns für den Erfolg unserer KI sehr wahrscheinlich erschienen. Das spiegelt sich auch in unserer Gruppenaufteilung wieder: vier Leute wurden der Programmierung zugeteilt und jeweils nur eine Person für die Website, Animation und Dokumentation.

Nach der Verteilung der Aufgaben konnten wir dann langsam zu dem Kern der Sache vordringen und uns einen Gruppennamen geben. Dieser sollte allerdings in engem Zusammenhang mit dem Aussehen unseres Spaceballs und der Website stehen, was uns zu der Überlegung brachte, unter welches Motto wir unseren Spaceball stellen möchten. Für das Thema und die Figur wurden umfangreich Vorschläge gesammelt, alle notiert und schließlich abgestimmt und noch einmal genau überlegt, inwieweit sich diese Idee realisieren ließe. Durchgesetzt hat sich schließlich eine Raupe, namens KillerRaupi zu animieren, die sich in einer niedlichen natürlichen Umgebung bewegen sollte, was auch unsere Website charakterisieren wird. Daraus leitete sich dann schlussendlich auch der Teamname unserer Gruppe ab: KillerRaupi.

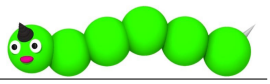
Als weiterer wichtiger Punkt der Tagesordnung stand die Klärung der Kommunikationswege an. Da nicht jeder aus der Gruppe ein Facebookkonto zu haben pflegte einigten wir uns auf einen Emailverteiler und eine WhatsApp-Gruppe zum Absprechen von Terminen und allgemeinen Belangen.

Den letzten Punkt bildete die Eintragung unserer Gruppe im Onlineportal und die Terminplanung für die kommende Woche. Ein neues Treffen wurde auf den 15.10.2014 angesetzt, bei dem der allgemeine Fortschritt in Programmierung, Dokumentation, Animation und Website geklärt und jeder somit geupdated werden soll. Des Weiteren muss beschlossen werden, ob diese Gruppenaufteilung sinnvoll ist, ob Inhalte noch spezifiziert werden müssen oder ob wie geplant verfahren und gearbeitet werden kann.



3.2 2. Meeting - 15.10.2014

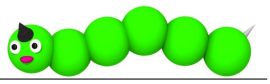
Die Tagesordnung des zweiten Gruppentreffens bestand hauptsächlich darin alle auf den aktuellen Stand des Projektfortschrittes zu bringen. Insgesamt waren damit alle soweit sehr zufrieden.
Als gewisses Problem stellten sich die Absprachen innerhalb der Gruppe heraus, deshalb wurde auch hier noch einmal über ein besseres System diskutiert. Daraus ergab sich, dass es sich doch bewährt, wenn man innerhalb der Programmierung der KI noch Untergliederungen vornimmt. Zum Beispiel beschäftigt sich nun Markus Funke im Detail mit der Optimierung des Tankprozesses, Max Landeg mit einem verbesserten Abbrems- und Antiwand-Kollisions-System und Torben Koch im Detail mit Angriff und Verteidigung des Spaceballs.
Des Weiteren wurde beschlossen, dass die Dokumentation in Grundzügen von den Programmierungen der Website und der Animation selbst vollzogen wird, da es sich als relativ unmöglich herausstellte, dass eine Person für die Dokumentation die Übersicht über alle Programme und Einzelfortschritte des Programmieres behält.
Der letzte Punkt der Tagesordnung bestand darin, sich auf Termine zu einigen, an denen wir die vorgeschriebenen Minipräsentationen abhalten werden, wobei dies auf den nächsten Mittwoch festgesetzt wurde.
Weiterhin zu diesem Treffen wurde, wenn jemand mit seiner Arbeit nicht weiter kam, die gesamte Gruppe zur Problemlösung zu Rate gezogen und intensiv weitergearbeitet.



3.3 3. Meeting - 22.10.2014

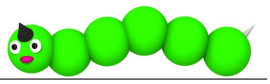
Das Gruppentreffen dieser Woche war dominiert von den ersten Minipräsentationen bei unserem Dozenten. Jeder hatte sich in den vergangenen Tagen überlegt was er präsentieren wollte und auch entsprechende Erklärungen vorbereitet.
Nachdem alle ihr Thema vorgestellt hatten, werteten wir innerhalb der Gruppe die durch den Dozenten geübte Kritik und vorgeschlagene Verbesserungsvorschläge aus und verteilten sie auf die einzelnen Personen. Beispielsweise wurde vorgeschlagen im Code der KI mit „flags" zu arbeiten, da diese lange Passagen vereinfachen können und somit Rechenaufwand einzusparen vermögen.
Weiterhin wurde uns nahe gelegt, die Dokumentation nicht gar so ausführlich zu gestalten, was wir natürlich gerne annehmen und ab dieser Woche auch so gestalten werden.

Des Weiteren wurde in diesem Meeting noch das Aussehen von KillerRaupi besprochen. Nach dem derzeitigen Stand war sie irgendwie noch nicht niedlich genug und so wurde in der Gruppe beschlossen, dass KillerRaupi größere Kulleraugen, sowie anstatt des glatten Horns, ein gezwirbeltes, wie ein Einhorn, erhalten sollte. Das wurde von den Animatoren auch sofort umgesetzt. So kann die neue KillerRaupi nun die Website und die Dokumentation zieren.
Weiterhin wurde Organisatorisches diskutiert, besonders was die Übermittlung der Dokumentationen der einzelnen Teilbereiche anging.
Da wir bereits sehr gut voran gekommen waren (der anfängliche Enthusiasmus), beschlossen wir auch von nun an in ein wenig gediegenerem Tempo weiter zu arbeiten, da noch ausreichend Zeit vorhanden ist.



3.4 4. Meeting - 5.11.2014

Nachdem es in der vergangenen Projektwoche keinen Bedarf für und folglich dann auch kein Gruppentreffen gegeben hatte, war die Gruppe diese Woche wieder reif für ein Treffen.
Wir haben gemeinsam die einzelnen Erfahrungen der letzten Projektwochen ausgetauscht, im Besonderen die Erkenntnisse aus den Minipräsentationen ausgewertet. Dahingehend haben wir den Weg für eine neue Strategie unserer KI ausgelotet und beschlossen ihr irgendwie beizubringen zu planen und optimale Fahrwege auszurechnen. Das wird in den folgenden Wochen dann auch der Kerninhalt der Aufgabe der Programmierung sein müssen.
Weiterhin haben wir uns über das visuelle Erscheinungsbild von Dokumentation, Website und Animation ausgetauscht, da das Visuelle ja häufig geschmackssache ist und man Entscheidungen diesbezüglich mit der Gruppe abgleichen sollte. So ist das angepasste Logo unserer Raupe, die in der Kopfzeile dieser Dokumentation auf der Linie entlang kriecht entstanden, sowie neue Bilder für KillerRaupis Bildergalerie auf unserer Website.
Des Weiteren hat jedes Mitglied die Gruppe auf den aktuellen Stand seiner Arbeit gebracht und nötige Absprachen für den reibungslosen Ablauf aller Arbeiten getroffen.

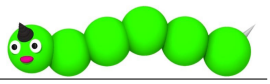


3.5 5. Meeting - 12.11.2014

Dieser Veranstaltung wurde der Titel „Jahreshauptversammlung" verliehen. Das liegt zum einen an einer amüsierten Gruppe, zum anderen fehlten einige Gruppenmitglieder sodass die Anwesenden sich genötigt fühlten, die Relevanz dieses eigentlich „normalen" Treffen ein wenig zu steigern, was also zu dieser Sonderbezeichnung führte.

Nach mehr als 5 Wochen getrennter Arbeit innerhalb der Programmierung erachteten wir es für notwendig endlich alle Teilbereiche (Tanken und Antiwandsystem, Angriff) einmal zusammenzufügen und herauszufinden, wie unser Spaceball sich in der Gesamtperformance gibt. Wir durften feststellen, dass das Tanken mit allen seinen Teilbereichen - Mittelpunkt, Winkelberechnung, Korrektur, Abbruchbedingung, Normal-Tanken und Anhaltesystem - sehr gut funktioniert. Die Minenabwehr funktioniert unschön und ist schleunigst verbesserungswürdig, funktioniert aber erst einmal. Das Antiwand-System funktioniert wie bisher sehr gut. Der Angriff wurde nach wenigen Simulationen doch heraus gelassen, da er nur sehr bedingt funktioniert.

Somit hatten wir ermittelt was noch verbesserungswürdig ist, aber auch worauf wir erst einmal stolz sein können.



3.6 6. Meeting - 19.11.2014

Heute ging es um Fehlerkorrektur und einen Schreibplan. Gemeinsam haben wir einige Texte der Website Korrektur gelesen und Grammatikfehler in der Dokumentation ausgemerzt. Weiterhin haben wir besprochen was in die Beschreibungen zu den einzelnen Teammitgliedern hinein gehört und uns auf Name, Foto, Arbeitsbereich und ein Motto geeinigt.
Markus und Max haben weiterhin noch einmal gemeinsam an einem besseren Minensystem gearbeitet.
Die beiden haben hierbei außerdem eine sehr interessante Entdeckung gemacht. Aufgrund der Brisanz dieser Entdeckung mussten leider alle ausschlaggebenden Wörter durch den Namen unserer Entdeckung „hexhex" ersetzt werden.

Unsere hexhex besteht darin, dass hexhex, oder besser gesagt das hexhex bzw. hexhex, ein hexhex bekommt, wenn die hexhex aus dem hexhex hexhex lautet. Dann greift ein hexhex kleiner hexhex, der den hexhex in der hexhex im hexhex hexhex lässt. Das ist insofern natürlich recht hexhex, da man so vom hexhex nicht mehr hexhex werden hexhex und dementsprechend, sofern hexhex besser hexhex kann als der hexhex, immer einen hexhex in der hexhex sicher hat. Somit wäre es rein hypothetisch möglich hexhex als hexhex in einer hexhex zu setzen und somit als letzten hexhex zu behalten. Da das allerdings extrem hexhex wäre, kommt es hexhex nicht in hexhex, daraus ein hexhex zu machen und hexhex zu nutzen.

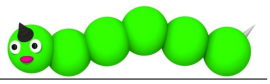
So und jetzt im Ernst:
Unsere Erkenntnis besteht darin, dass „Spaceballs", oder besser gesagt das m-file bzw. Matlab, ein Problem bekommt, wenn die Endausgabe aus dem m-file „Not-a-Number" lautet. Dann greift ein netter kleiner Bug, der den Spaceball in der Visualisierung im Browserfenster verschwinden lässt. Das ist insofern natürlich recht praktisch, da man so vom Gegner nicht mehr berührt werden könnte und dementsprechend, sofern man besser Tanken kann als der Gegner, immer einen Punkt in der Gesamtwertung sicher hat. Somit wäre es rein hypothetisch möglich Not-a-Number als Befehl in einer flag zu setzen una somit als letzten Notnagel zu behalten. Da das allerdings extrem unsportlich und schlichtweg Betrug wäre, kommt es überhaupt nicht in Frage, daraus ein System zu machen und ernsthaft zu nutzen.

3.7 7. Meeting - 3.12.2014

Bei diesem Meeting war es wieder einmal nötig einander auf den aktuellen Stand des Fortschritts zu bringen (schließlich liest sich nicht jeder jede Woche die Dokumentation durch). In der Animation bedeutete das, dass recht viele Videos fertig gestellt worden waren, die alle gesichtet werden mussten.
Weiterhin haben Johanna und Timo noch einmal gemeinsam an einem verbesserten Logo gefeilt und es dann auch entsprechend eingearbeitet.
Innerhalb des Teams wurde auch noch einmal über KillerRaupis Erscheinungsbild diskutiert und festgestellt, dass Raupi ohne Füße eigentlich wie ein Wurm aussieht. Daher soll sie noch kleine Beinchen und Füße erhalten. Das stimmt die Animationsfraktion natürlich nicht besonders froh, da sie dann noch einmal die Videos und Fotos überarbeiten müssten. Trotzdem soll diese Idee in die Tat umgesetzt werden.
Des Weiteren saßen Max und Markus wieder einmal gemeinsam am Minensystem, das die beiden wirklich noch bis zur Perfektion verbessern wollen. Es geht auch mit sehr großen Schritten vorwärts.

Weiterhin ist uns mehr oder weniger aufgefallen, dass wir immer noch Portraits von uns für die Website und die Dokumentation und eben im allgemeinen brauchten. Um diesen Portraits ein wenig mehr Individualität zu verleihen, haben wir uns überlegt die langweiligen Originalportraits mithilfe von Bildbearbeitung in Cartoons zu verwandeln. Schnell wurde ein entsprechendes Programm gesucht und gefunden, aber da die Portraits auf der Website durchaus der Öffentlichkeit präsentiert werden sollten, bahnten sich bereits wieder rechtliche Probleme an. Wir mussten erst einmal so verbleiben, dass wir die Anbieter des Programms kontaktieren, ihnen den Sachverhalt schildern und hoffentlich einen Freifahrtschein erhalten werden. Von der grundsätzlichen Idee aber wollen wir nicht mehr abweichen.

Außerdem haben wir als Team eine kleine Weihnachtsfeier geplant und durchgeführt, denn schließlich soll der Spaß auch nicht zu kurz kommen und gemeinsame Unternehmungen stärken auch den allgemeinen Zusammenhalt und die Integration in das Team selbst, sowie die persönliche Definition mit dem Projekt.



3.8 8. Meeting - 10.12.2014

Diese Projekttreffen war relativ kurzfristig angesetzt worden und hatte sich mehr oder weniger spontan ergeben, da sehr viele Teammitglieder eine Minipräsentation halten mussten. Diese wurden dann entsprechend beim Gruppentreffen ausgewertet, sowie Verbesserungsvorschläge angenommen aber auch abgelehnt.

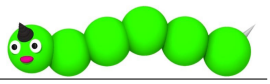
Von den Anbietern des Bildbearbeitungsprogrammes zum Erstellen unserer Cartoons gab es bedauerlicher Weise noch keine Rückmeldung. So war in dieser Problematik leider noch kein Fortschritt erzielt worden.

Der weitere Verlauf unseres Gruppentreffens war sehr demokratisch geprägt, denn es galt eine Entscheidung bezüglich der Mauer, gegen die unsere KillerRaupi im Video „Bande" fahren sollte, zu treffen. Unser fleißiges Allround-Talent Timo hatte bereits 3 Entwürfe vorbereitet und nun musste theoretisch nur noch der schönste ausgewählt werden. Eine Abstimmung ergab dass es jeweils 2 Stimmen für jeden Mauertyp gab und eine Enthaltung, wir waren also genau so schlau wie vor der Wahl. Irgendwie hat es sich durch darauffolgende Diskussionen dann doch so ergeben, dass eine Mauer ausgewählt werden konnte mit der schließlich alle einverstanden waren...

3.9 9. Meeting - 16.12.2014

Das letzte Treffen in diesem Jahr und vor der Weihnachtspause. Diese Tatsache war es auch geschuldet, dass das Gruppentreffen an einem Dienstag - und nicht wie sonst üblich an einem Mittwoch - stattfand (am Mittwoch waren zu viele Mitglieder bereits auf dem nach-Hause-Weg).
Trotz der ungewohnten Zeit konnten wir sehr produktiv arbeiten, denn es ging um die bereits lange angekündigten und angedrohten Fotos in Portraitaufnahme. Jeder musste sich einmal an die Wand stellen und durfte mehr oder weniger nett schauen um anschließend seinen Abzug bewundern zu können. Nachdem alle einigermaßen zufrieden mit ihrer Aufnahme waren und sich auf dem Bild wiedererkannt hatten, konnten die Fotos an das bearbeitende Teammitglied der Animation vertrauensvoll übergeben werden.

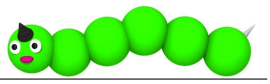
Von dem Anbieter des Fotobearbeitungsprogrammes hatten wir auch in jüngster Vergangenheit noch keine Rückmeldung erhalten, was uns dazu verleitete uns einmal an den betreuenden Professor zu wenden und diesbezüglich Erkundigungen einzuziehen. Dieser konnte uns leider auch nur mitteilen, dass, wenn wir das Programm trotzdem verwenden und die bearbeiteten Fotos online stellen, wir das auf unser eigenes Risiko tun. Das heißt unterm Strich waren wir leider immer noch nicht schlauer, auch wenn wir nun schon einmal die Rohlinge der Fotos haben.



3.10 10. Meeting - 07.01.2015

Nach der zweiwöchigen Weihnachtspause war es nun Zeit wieder einmal in der Gruppe unser Projekt vorantreiben.
Wir haben gemeinsam zusammengefasst, welche Aufgaben es noch zu erledigen gibt und diese verteilt. Drei Leute von uns müssen sich beispielsweise noch um die Sounds für unsere Videos kümmern, damit alle Clips noch bis zum Turnier pünktlich fertig werden.
Des Weiteren haben sich Dokumentation und Website noch einmal über die gemeinsamen Texte ausgetauscht, sowie die Porträtfotos in angebrachte Größen zugeschnitten und entsprechend eingefügt.

Ein großes gemeinsames Gruppentreffen wird es wohl nicht noch einmal geben, dafür besteht kein Bedarf mehr. Vor allem einzelne Mitglieder deren Aufgabenbereiche einander tangieren werden sich einzeln noch zusammen setzen müssen. Die Hauptaufgabe der Programmierer wird dabei natürlich sein, alle einzelnen Teilbereiche unserer kraft-Datei zusammen zu bringen und die einzelnen Systeme feinzujustieren.



4 Fortschritte in der Programmierung der KI

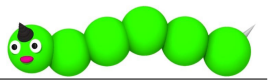
4.1 1. Projektwoche

Grundaufgabe der Programmierung besteht darin das m-file „kraft" zu erstellen, in welchem der Schubvektor f von KillerRaupi je nach Tätigkeit - Tanken, Angriff, Verteidigung - bestimmt wird.
Demzufolge bekommt die function in Matlab als Eingangsparameter die Farbe des zugehörigen Spaceballs und den Aufbau und Inhalt des Spielfeldes mitgeteilt und gibt den Schubvektor aus (Zeile 1). Ein dazugehöriges Video wird automatisch mit ausgegeben.

```
1  function [ f, video ] = kraft( spielfeld, farbe )
2
3  % Weist ich und gegner jeweils rot oder blau zu (ist erforderlich)
4  if strcmp (farbe, 'rot')
5      ich=spielfeld.rot;
6      gegner=spielfeld.blau;
7  else
8      ich=spielfeld.blau;
9      gegner=spielfeld.rot;
10 end
11
12 % Dieses f schickt unsere KillerRaupi nach rechts
13 f=[1 0];
14
15 % Diese Zuweisungen sind erforderlich (nicht loeschen) :)
16 video.index=0;
17 video.text='';
18
19 % Regelt die Hoechstgeschwindigkeit
20 if(norm(ich.ges)>0.15)
21     f=[0 0];
22 end
23
24 end
```

Wie im vorangegangen Code zu sehen ist, wird in Zeile 3 bis 10 zugewiesen, um welchen Spaceball mit welcher Farbe es sich bei unserem handelt.
Die nächste Zuweisung in Zeile 13 dient einzig eines Testlaufes, ob soweit alles funktioniert. Hierbei wird der Schubvektor so definiert, dass KillerRaupi nach rechts fährt.
Die if-Bedingung beginnend in Zeile 20 war zunächst als äußerst trickreicher Höchstgeschwindigkeitslimiter gedacht, der ab einer bestimmten Geschwindigkeit den Schubvektor auf 0 reduziert. Diese Spielerei entpuppte sich im weiteren Programmierungsverlauf jedoch als recht unnötig.

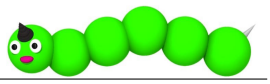
Das wesentlichste Kriterium für einen intelligenten und erfolgreichen Spaceball scheint uns das Tanken, weshalb wir in dieser ersten Projektwoche uns ausschließlich mit dem Ansteuern, Treffen und Leertanken der Tankstellen befassen. Um hier unseren Fortschritt besser abschätzen zu können verringern wir im m-file „spiel" die Anzahl der Minen auf Null, um uns ungestört dem Tanken widmen zu können. Der Code für das



Tanken ist im Folgenden zu sehen:

```
25 % Tanken aktiviert den "Tank-modus", bei 0 kein tanken
26 tanken=1;
27
28 aktuelleTanke=spielfeld.tanke(1);
29
30
31 for i=1:10
32     tankei=spielfeld.tanke(i);
33     if(norm(ich.pos-tankei.pos)<norm(ich.pos-aktuelleTanke.pos) &&
           tankei.radius>0.02)
34         aktuelleTanke=spielfeld.tanke(i);
35     end
36 end
37
38 if(tanken==1)
39         alpha=acos(dot(ich.ges, aktuelleTanke.pos-ich.pos)/(norm(ich.ges)*
               norm(aktuelleTanke.pos-ich.pos)));
40
41     if (norm(ich.pos-aktuelleTanke.pos)<(ich.radius+aktuelleTanke.radius))
42         f=-ich.ges;
43         if(norm(ich.ges)<0.02)
44             f=[0 0];
45         end
46         video.text='Stop!';
47     elseif(abs(alpha)<1 && norm(ich.pos)>0.1 && norm([1 0]-ich.pos)>0.1 &&
           norm(ich.ges)>0.12)
48         f=[0 0];
49     else
50         f=aktuelleTanke.pos-ich.pos;
51     end
52
53 end
54
55 end
```

Zuerst wird der Tank-Modus in Zeile 26 aktiviert. Dies geschieht nur erst einmal vorsorglich, schließlich gibt es noch keine alternativen Modi, aber wie genau die Umschaltung zwischen den Modi funktionieren soll, ist erst einmal noch nicht Gegenstand unserer Überlegungen gewesen. Deshalb wird vorsorglich der Tank-Modus mit erwähnt.
In Zeile 28 wird die im Spielfeld mit der Zahl 1 bezeichnete Tankstelle als unsere aktuelle und für KillerRaupi interessante Tankstelle voreingestellt. Das ist natürlich nicht besonders effektiv, da diese Tankstelle überall und auch sehr klein sein kann. Deswegen wird in der ab Zeile 31 beginnenden Schleife eine für KillerRaupi attraktivere Tankstelle gesucht.
Im ersten Teilschritt der folgenden if-Bedingung (Zeile 38) wird mittels Skalarprodukt der Winkel zwischen dem aktuellen Geschwindigkeitsvektor von KillerRaupi und dem Richtungsvektor von KillerRaupi zur aktuellen Tankstelle berechnet. Dieser Winkel alpha wird im übernächsten Schritt noch gebraucht.
In Zeile 41 wird definiert, wie die KI reagieren soll, wenn sich die beiden Radien - Radius Tankstelle und Radius Spaceball - schneiden. Dann soll KillerRaupi als Schubvektor die negative derzeitige Geschwindigkeit erhalten, also quasi den Umkehrschub an-

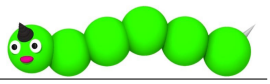


schalten. Sollte der Betrag der derzeitigen Geschwindigkeit allerdings unter einem bestimmten Wert liegen -in diesem Fall handelt es sich um einen willkürlich ausgewählten Wert- dann wird der Schubvektor auf 0 gesetzt.
In beiden Fällen wird als Videountertitel „Stop!" eingeblendet.
Im folgenden Teilschritt wird nun der Winkel alpha verarbeitet. Wir hielten es für sehr sinnvoll, dass, wenn KillerRaupi nahezu den idealen Kurs hat um die Tankstelle zu erreichen, keine Kursänderung mehr vorgenommen werden muss, schließlich würde das nur wertvollen Sprit verbrauchen. Diese Option setzt sich aus vier Bedingungen zusammen, die alle erfüllt sein müssen: der Betrag von alpha muss kleiner als 1 sein; die Position muss größer als 0.1 sein, das bedeutet, dass er nicht in einer Spielfeldecke stehen soll und die 3. Bedingung beinhaltet die selbe Funktion für eine andere Ecke. Schlussendlich und als letzte Bedingung muss die Geschwindigkeit von KillerRaupi größer als 0.12 sein. Wenn das erfüllt ist, dann wird der Schubvektor f wieder auf 0 heruntergefahren und somit an die Tankstelle angedockt.

```
57 if(tanken==1)
58     alpha=acos(dot(ich.ges, aktuelleTanke.pos-ich.pos)/(norm(ich.ges)*norm(
           aktuelleTanke.pos-ich.pos)));
59
60     if (norm(ich.pos-aktuelleTanke.pos)<(ich.radius+aktuelleTanke.radius) ||
           (norm(aktuelleTanke.pos-ich.pos)<0.10 && norm(ich.ges)>0.1))
61         f=-ich.ges;
62
63         video.text='Stop!';
64     elseif(abs(alpha)<3 && norm(ich.pos)>0.1 && norm([1 0]-ich.pos)>0.1 &&
           norm(ich.ges)>0.12)
65         f=[0 0];
66         video.text='Auf Tanke zu fliegen';
67     else
68         f=aktuelleTanke.pos-ich.pos;
69         video.text='Zielen!';
70     end
71
72 end
```

Als kleine Spielerei haben wir als nächstes verschiedene Videountertitel für die Hinfahrt zur Tankstelle erstellt, der Code ist in der vorangegangen Abbildung zu sehen. Wenn KillerRaupi vor der Tanke zum Stehen kommen soll und den Umkehrschub aktiviert, erscheint „Stop!". Wenn KillerRaupi schon einen guten Tankkurs gefunden hat, wird „Auf Tanke zu fliegen" eingeblendet und wenn die KI eine neue Tankstelle ansteuern soll wird „Zielen!" angezeigt.
Des Weiteren wurde an dieser Stelle noch erneuert, dass KillerRaupi eher zu bremsen beginnt. Das wird mithilfe einer if-Bedingung geregelt. Wenn die KI merkt, dass sie bereits auf 0.15 an die Tankstelle heran ist, wird der Umkehrschub aktiviert. Das machte den Bremsvorgang schon erheblich eleganter, funktioniert jedoch nicht, wenn der Spaceball sehr groß und schnell ist, dann kam er einfach nicht zum stehen. Hier muss auf jeden Fall eine Berechnung her, die den Schub regelt.

Eine offene Baustelle stellt bis dahin auch die Bande dar. KillerRaupi fuhr zwar schon sehr gezielt die Tankstellen an, wenn diese allerdings zu nahe am Rand lagen und der Spaceball beim Tanken zu groß wurde, stieß sie mehrmals mit der Bande zusam-



men. Um dieses Problem zu lösen, führten wir für jede Wand einen Puffer ein und packten diese in mehrere if-Bedingungen, wie im untenstehenden Code zu sehen ist.

Je nach Geschwindigkeit und Richtung mit der auf eine Wand zugefahren wird, wird hierbei eine Gegenmaßnahme definiert, mit der KillerRaupi sich dann von der Wand weg manövrieren soll. Die Puffer an sich wurden willkürlich mit uns passend scheinenden Zahlen gefüllt, wie in der nebenstehenden Abbildung zu sehen ist.

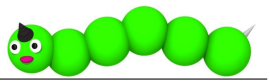
```
73 if(ich.radius<0.05)
74     pufferRand1=0.05;
75     pufferRand2=0.1;
76 elseif(ich.radius<0.1)
77     pufferRand1=0.2;
78     pufferRand2=0.3;
79 else
80     pufferRand1=0.3;
81     pufferRand2=0.4;
82 end
```

Diese Programmierung hindert KillerRaupi zwar meistens daran in eine Wand hinein zu fahren, allerdings versperrte es auch den Weg zu am Rand liegenden Tankstellen.

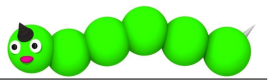
Weiterhin bleibt das Problem bestehen, dass wenn KillerRaupi zu behäbig und zu schnell war, sie es nicht mehr schaffte vor einer Wand zu bremsen. Auch hier ist es nötig Berechnungsvorschriften zu erstellen, die einen korrekten Bremsweg quasi ausrechnen, damit KillerRaupi immer früh genug noch abdrehen kann.
Um das Suchen einer Tankstelle noch zu optimieren haben wir eine neue Funktion geschrieben, die darauf spezialisiert ist eine aktuelle Tanke, die angefahren werden soll, zu finden. Der Code dazu ist in der folgenden Abbildung zu sehen.

```
83 function [ aktuelleTanke ] = aktuelleTankeBestimmen (spielfeld, ich)
84 aktuelleTanke=spielfeld.tanke(randi([1 10],1,1));
85
86
87 for i=1:10
88     tankei=spielfeld.tanke(i);
89     if(norm(ich.pos-tankei.pos)<norm(ich.pos-aktuelleTanke.pos) &&
           tankei.radius>0.02)
90         aktuelleTanke=spielfeld.tanke(i);
91     end
92 end
93 disp('aktuelleTankebestimmen')
94
95
96 end
```

Diese function erhält als Eingangsparameter die Daten vom Spielfeld und dem Spaceball und gibt als Ausgangsparameter die aktuelle interessante Tanke aus. Zu diesem Zweck wird erst einmal per Zufallsprinzip (Zeile 83f) irgendeine Tanke auf dem Feld als aktuelle Tanke voreingestellt. Dann wird in einer for-Schleife für alle 10 Tankstellen geprüft, ob sie am nächsten an der derzeitigen Position des Spaceballs dran sind und eine Mindestgröße von 0.02 überschreiten. Sind beide Bedingungen erfüllt, dann begibt sich KillerRaupi zur nächsten Tankstelle. Überdies wird mit der Zeile 93 im Command Window „aktuelleTankebestimmen" ausgegeben, da von uns auch von Interesse war zu sehen, wie oft diese Schlaufe und ob überhaupt durchlaufen wird.



Das funktioniert auch zur vollsten Zufriedenheit.
Als Zusammenfassung dieser Woche kann man also sagen, dass das Tanken soweit schon sehr gut funktioniert.
Verbessert werden muss unbedingt das Abbremsverhalten, hierbei sollen zuverlässigere Werte aus Berechnungen entnommen werden. Damit soll auch die Annäherung an die Wände zuverlässiger funktionieren. Noch keine Funktion haben wir für Angriff und Verteidigung gefunden, doch das soll jetzt auch erst einmal noch nicht im Vordergrund stehen. KillerRaupi soll erst einmal mit allen Herausforderungen des Spielfeldes klar kommen.



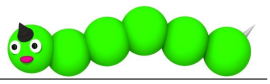
4.2 2. Projektwoche

Wie in der letzten Woche vorgenommen, haben wir das Antiwand-System verbessert. Für alle Seiten der Bande (rechts,links,oben,unten), sowie die Ecken des Spielfeldes existiert nun ein recht funktionales Wandpräventationssystem. Die folgende Abbildung zeigt den ersten Teil davon.

```
1  %% Wandpraeventionssystem
2  if (ich.ges(1)>0 && abs(ich.ges(1))>sqrt(2*(4e-5/(ich.radius)^2)*(1-ich.pos
       (1)-ich.radius-0.01))) || 1-ich.pos(1)<ich.radius+0.01
3       % rechts
4       f=[-1 0];
5       video.text='rechte Bande';
6  end
7  if (ich.ges(1)<0 && abs(ich.ges(1))>sqrt(2*(4e-5/(ich.radius)^2)*(ich.pos(1)-
       ich.radius-0.01))) || ich.pos(1)<ich.radius+0.01
8       % links
9       f=[1 0];
10      video.text='linke Bande';
11 end
12 if (ich.ges(2)<0 && abs(ich.ges(2))>sqrt(2*(4e-5/(ich.radius)^2)*(ich.pos(2)-
       ich.radius-0.01))) || ich.pos(2)<ich.radius+0.01
13      % oben
14      f=[0 1];
15      video.text='obere Bande';
16 end
17 if (ich.ges(2)>0 && abs(ich.ges(2))>sqrt(2*(4e-5/(ich.radius)^2)*(1-ich.pos
       (2)-ich.radius-0.01))) || 1-ich.pos(2)<ich.radius+0.01
18      % unten
19      f=[0 -1];
20      video.text='untere Bande';
21 end
```

Hier wird für jede mögliche Kollisionsposition mit der Wand eine if-Bedingung durchlaufen. In dieser wird jeweils verglichen, ob die x-, bzw. y-Komponente des Geschwindigkeitvektors „ich.ges" grundsätzlich von Null verschieden ist, also ob der Spaceball sich bewegt und auf die Wand zusteuert. Wenn dieser Wert größer ist als der Bremsweg, den der Spaceball bis zur Wand braucht, welcher hier mithilfe der Bewegungsgleichungen berechnet wird, dann muss der Spaceball natürlich bremsen.
Wenn diese if-Bedingung erfüllt wird und der Spaceball praktisch auf die Bande zusteuert, dann wird ein Schub eingestellt, der $90°$ von der Wand weg zeigt.
Der Doppelstrich fast am Ende der Code-Zeilen 149,154,159 und 164 schließt noch eine Oder-Bedingung an. Mit dieser soll vermieden werden, dass der Spaceball wenn er auf dem Weg Richtung Wand ist um zu einer Tankstelle zu gelangen, beim Tanken zu groß und zu behäbig wird, was den Bremsweg verlängern und der Spaceball somit doch in die Bande fliegen würde. Dieses „oder" führt also dazu, dass eher gebremst wird, wenn der Radius des Spaceballs vergrößert wird.
Zu jedem Fall wird neben dem Video auch ein entsprechender Text eingeblendet, was in den Zeilen 152,157,162 und 167 gecoded wurde.

Der zweite Teil des Wandpräventationssystems beschäftigt sich mit der Kollision in den Ecken der Bande. Hier müssen pro möglicher Ecke jeweils x- und y-Komponente



des Geschwindigkeitsvektors „ausgewertet" werden um zu sehen, ob sich der Spaceball tatsächlich auf die jeweilige Ecke zubewegt. Das findet natürlich auch wieder mithilfe von if-Bedingungen statt. Wenn die Komponenten von Null verschieden sind und der Bremsweg größer als der Betrage des Geschwindigkeitsvektor ist, oder sich der Radius mehr vergrößern würde, als der Abstand zur Wand beträgt, dann wird ein entsprechender Gegenschubvektor f ausgegeben, der genau in die entgegengesetzte Richtung der der Ecke weist.
Das beschriebene Vorgehen ist in Abbildung o aus dem kraft-file ebenfalls herauszulesen.

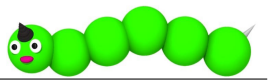
```
22 if ((ich.ges(1)>0 && abs(ich.ges(1))>sqrt(2*(2e-5/(ich.radius)^2)*(1-ich.pos
       (1)-ich.radius-0.01)))&&(ich.ges(2)<0 && abs(ich.ges(2))>sqrt(2*(2e-5/(
       ich.radius)^2)*(ich.pos(2)-ich.radius-0.01))))
23     % oben rechts
24     f=[-1 1];
25 end
26 if ((ich.ges(1)>0 && abs(ich.ges(1))>sqrt(2*(2e-5/(ich.radius)^2)*(1-ich.pos
       (1)-ich.radius-0.01)))&&(ich.ges(2)>0 && abs(ich.ges(2))>sqrt(2*(2e-5/(
       ich.radius)^2)*(1-ich.pos(2)-ich.radius-0.01))))
27     % unten rechts
28     f=[-1 -1];
29 end
30 if ((ich.ges(1)<0 && abs(ich.ges(1))>sqrt(2*(2e-5/(ich.radius)^2)*(ich.pos(1)
       -ich.radius-0.01)))&&(ich.ges(2)>0 && abs(ich.ges(2))>sqrt(2*(2e-5/(
       ich.radius)^2)*(1-ich.pos(2)-ich.radius-0.01))))
31     % unten links
32     f=[1 -1];
33 end
34 if ((ich.ges(1)<0 && abs(ich.ges(1))>sqrt(2*(2e-5/(ich.radius)^2)*(ich.pos(1)
       -ich.radius-0.01)))&&(ich.ges(2)<0 && abs(ich.ges(2))>sqrt(2*(2e-5/(
       ich.radius)^2)*(ich.pos(2)-ich.radius-0.01))))
35     % oben links
36     f=[1 1];
37 end
```

Ebenso wurde der Prozess des Tankens noch einmal neu aufgearbeitet. Zuerst wird eine aktuelle Tankstelle bestimmt.

```
38 % auswaehlen der aktuellen Tanke
39 aktuelleTanke=aktuelleTankeBestimmen(spielfeld, ich);
40
41 % wenn Tanke zu klein, wieder neue aktuelle Tanke bestimmen
42 if(aktuelleTanke.radius<0.005)
43     aktuelleTanke=aktuelleTankeBestimmen(spielfeld, ich);
44 end
```

Dies geschieht mit der letzte Woche erstellen und diese Woche verbesserten function „aktuelleTankeBestimmen", welche im Folgenden dargestellt ist.

```
45 function [ aktuelleTanke ] = aktuelleTankeBestimmen (spielfeld, ich)
46 aktuelleTanke=spielfeld.tanke(randi([1 10]));
47
48 for i=1:10
49     tankei=spielfeld.tanke(i);
50     if(norm(ich.pos-tankei.pos)-8*tankei.radius<norm(ich.pos-
           aktuelleTanke.pos)-8*aktuelleTanke.radius && tankei.radius>0.01)
```



```
51          aktuelleTanke=spielfeld.tanke(i);
52      end
53  end
54  end
```

Diese function legt durch Zufall, das geschieht durch den Befehl „randi", eine aktuelle Tankstelle fest, die sich auf dem Spielfeld befindet. Diese zufällig ausgewählte Tankstelle wird dann innerhalb der for-Schleife mit den anderen Spielfeld-Tankstellen verglichen. Die Kriterien dabei sind, welche Tankstelle am nächsten an der derzeitigen Position des Spaceballs liegt und dabei noch einen Radius von größer als 0.01 besitzt. Erfüllt die Tankstelle alle diese Kriterien, dann wird sie die aktuelle Tankstelle, mit der dann weiter gearbeitet werden kann.

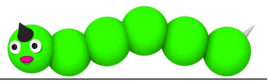
Wenn eine Tankstelle eigentlich leer getankt ist und es sich somit nicht lohnt erst dorthin zu fliegen, dann soll direkt eine neue Tankstelle bestimmt werden.

Danach schließt sich ein sehr umfangreiches Verfahren an, das bestimmt, ob 2 Tankstellen so nahe beieinander liegen, dass es sinnvoll ist den Mittelpunkt zwischen den beiden Tankstellen anzusteuern und somit beide auf einen Schlag leerzusaugen, was unserem Spaceball sprit-technisch einen enormen Vorteil verschaffen würde.

```
55  if(tanken==1)
56      system=1;
57      if norm(aktuelleTanke.pos-spielfeld.tanke(1).pos)<1.8*ich.radius+
            aktuelleTanke.radius+spielfeld.tanke(1).radius && spielfeld.tanke(1)
            .radius>0.001 && norm(aktuelleTanke.pos-spielfeld.tanke(1).pos)>0
58          system=2;
59          vglTanke=spielfeld.tanke(1);
60      end
61  % ...fuer alle Tanken von 1 bis 10...
62      if norm(aktuelleTanke.pos-spielfeld.tanke(10).pos)<1.8*ich.radius+
            aktuelleTanke.radius+spielfeld.tanke(10).radius && spielfeld.tanke(10)
            .radius>0.001 && norm(aktuelleTanke.pos-spielfeld.tanke(10).pos)>0
63          system=2;
64          vglTanke=spielfeld.tanke(10);
65      end
```

In diesem Verfahren wird jede Tankstelle im Spielfeld mit der aktuell ausgewählten Tankstelle verglichen. Hierbei wird ermittelt, ob die Radien groß genug sind um beide Tankstellen gleichzeitig zu benutzen, dann wird die durchgeführt und System 2 angeschaltet, welches sich mit dem Anfahren des Mittelpunktes zwischen den Tankstellen beschäftigt. Der Radius wird in der Berechnung mit 1,8 multipliziert, da die Tankstelle sonst manchmal genau an der Grenze, oder zu klein ist, damit das Verfahren funktionieren kann. Des Weiteren wird abgeglichen, ob die aktuelle Tankstelle und die betrachtete Tankstelle identisch sind.
Wenn die Bedingungen erfüllt sind und 2 Tankstellen so nah beieinander liegen, dass sie gemeinsam leergesaugt werden können, dann wird System 2 angeschaltet.
Ist das nicht der Fall, dann bleibt System 1 angeschaltet, welches die Tankstelle dann „normal" und direkt anfährt.

```
66  alpha1=acosd(dot(ich.ges, aktuelleTanke.pos-ich.pos)/(norm(ich.ges)*norm(
        aktuelleTanke.pos-ich.pos)));
```

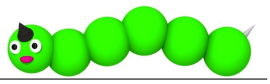


```
67      if system==2
68          alpha2=acosd(dot(ich.ges,(aktuelleTanke.pos+vglTanke.pos)/2-ich.pos)
                /(norm(ich.ges)*norm((aktuelleTanke.pos+vglTanke.pos)/2-ich.pos))
                );
69      end
```

Für System 1 wird nun der Winkel „alpha1“. Das ist der Winkel zwischen dem Geschwindigkeitsvektors des Spaceballs und dem Richtungsvektor der aktuellen Tankstelle. Wenn jedoch System 2 angeschalten ist, dann wird „alpha2“ berechnet, was der Winkel zwischen dem Geschwindigkeitsvektor des Spaceballs und dem Richtungsvektor des Mittelpunktes zwischen den 2 Tankstellen ist.

```
70      if (norm(ich.pos-aktuelleTanke.pos)<(ich.radius+aktuelleTanke.radius)) &&
            system~=2
71          f=-ich.ges;
72          if norm(ich.ges)<0.001 || norm(ich.pos-aktuelleTanke.pos)<0.01
73              f=[0 0];
74          end
75          video.text='Stop!';
76      elseif system~=2 && alpha1>10 && norm(ich.ges)>0.01
77          f=aktuelleTanke.pos-ich.pos-5*ich.ges;
78          video.text='Korrektur 1';
79      elseif system==2 && alpha2>15 && norm(ich.ges)>0.01 && norm((
            aktuelleTanke.pos+vglTanke.pos)/2-ich.pos)>0.02
80          f=((aktuelleTanke.pos+vglTanke.pos)/2)-ich.pos-5*ich.ges;
81          video.text='Korrektur 2';
82      elseif system==2 && aktuelleTanke.radius>0.01 && vglTanke.radius>0.01
83          f=((aktuelleTanke.pos+vglTanke.pos)/2)-ich.pos;
84          if norm((aktuelleTanke.pos+vglTanke.pos)/2-ich.pos)<0.01 && norm(
                ich.ges)<0.001
85              f=[0 0];
86          end
87          video.text='Mittelpunkt';
88      else
89          f=aktuelleTanke.pos-ich.pos;
90          video.text='Zielen!';
91      end
```

Nun folgt eine weitere if-Bedingung, wie in der vorangegangenen Abbildung zu sehen ist, welche sich nun mit dem konkreten Anfahren der gewünschten Punkte beschäftigt. Wenn der Spaceball die Tankstelle berührt, dann wird mit dem negativen Geschwindigkeitsvektor Gegenschub erzeugt und gebremst. Das wird allerdings nur ausgeführt, wenn System 2 nicht angeschaltet ist. Wenn man sich schon sehr nahe an der Tankstelle befindet, dann wird der Schub ganz auf Null geregelt und es wird im Video der Text „Stop“ eingeblendet.
Wenn eine Tankstelle direkt angefahren werden soll, der Kurs auf die Tankstelle jedoch um mehr als $10°$ abweicht, dann wird eine Korrektur vorgenommen. Bei dieser Berechnung wird der Geschwindigkeitsvektor stark mit eingerechnet.
Die folgenden Zeilen des Codes beziehen sich nun darauf, wenn tatsächlich der Mittelpunkt angefahren werden soll. Ist hier der Winkel „alpha2“ größer als $15°$, dann wird ebenfalls eine Korrektur vorgenommen und ein entsprechender Geschwindigkeitsvektor ausgegeben. Der letzte Teil der if-Bedingung beschreibt wie sich der Schubvektor f verhalten soll, wenn der Mittelpunkt normal angefahren werden soll. Dieser



ergibt sich dabei aus verschiedenen Berechnungen.

```
92  %% Anhaltesystem fuer Tanken
93  if norm(ich.ges)>=sqrt(2*(4e-5/(ich.radius)^2)*norm(ich.pos-aktuelleTanke.pos
        )) && tanken==1 && system==1
94       f=-ich.ges;
95       video.text='Anhalten!';
96  end
97
98  if system==2 && norm(ich.ges)>=sqrt(2*(4e-5/(ich.radius)^2)*norm(ich.pos-((
        aktuelleTanke.pos+vglTanke.pos)/2)))
99       f=-ich.ges;
100      video.text='Mittelpunkt-Anhalten';
101 end
```

Im vorangegangenen Code ist zu sehen, wie der Spaceball für System 1 oder 2 jeweils Anhalten soll. Die Berechnungsformel für diesen Schritt entspringt hierbei aus den physikalischen Bewegungsgleichungen. So können richtige Werte zum Vergleich herangezogen werden und nicht mehr nur willkürlich festgesetzte. In beiden Fällen wird dann als Schubvektor der negative Geschwindigkeitsvektor definiert, damit effektiv abgebremst werden kann.
Des Weiteren ist für den Tankprozess noch eine Abbruchbedingung notwendig, welche in der folgenden Abbildung zu sehen ist.

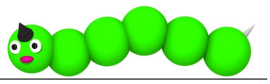
```
102 %% Abbruchbedingung
103 if spielfeld.tanke(1).radius<=0.01 && spielfeld.tanke(2).radius<=0.01 &&
        spielfeld.tanke(3).radius<=0.01 && spielfeld.tanke(4).radius<=0.01 &&
        spielfeld.tanke(5).radius<=0.01 && spielfeld.tanke(6).radius<=0.01 &&
        spielfeld.tanke(7).radius<=0.01 && spielfeld.tanke(8).radius<=0.01 &&
        spielfeld.tanke(9).radius<=0.01 && spielfeld.tanke(10).radius<=0.01
104      f=[0 0];
105      video.text='Ende';
106 end
```

Diese if-Bedingung dient dazu, dem Spaceball zu definieren, dass er stehen bleiben soll, wenn alle Tankstellen aufgebraucht sind, also einen Radius von kleiner 0,01 haben. Dann wird der Schubvektor auf Null geregelt.

Des Weiteren kann unsere KillerRaupi nun auch schon ein wenig angreifen und sich verteidigen.
Um die folgenden Bedingungen für den Angriff aufstellen zu können und einen „Angriffsplan" entwerfen zu können, muss zuerst der Winkel „beta" zwischen dem Geschwindigkeitsvektor unseres Spaceballs und des gegnerischen Spaceballs ermittelt werden, was mithilfe des Skalarproduktes zwischen 2 Vektoren berechnet wird.

```
107 % Winkel zwischen mir uns Gegner
108 beta=acosd(dot(ich.ges, gegner.ges)/(norm(ich.ges)*norm(gegner.ges)));
```

An diese Winkelberechnung schließt sich ein if-Bedingung an, die regeln soll, wie der Schubvektor für verschiedene Winkel β geregelt werden soll. Grundsätzlich wird als Bedingung gesetzt, dass der Radius unseres Spaceballs das 1,15-fache des Radius des Gegners betragen soll, sowie der eigene Radius größer als 0,04 sein soll.



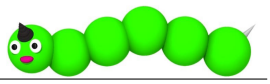
```
109 if ich.radius>1.15*gegner.radius && ich.radius>0.035 && norm(ich.pos-
        gegner.pos)<0.6
110      if 0<beta<15 || 180>beta>175
111      f=gegner.pos-ich.pos+gegner.ges+gegner.ges*2;
112      video.text='Angriff';
113      elseif 15<=beta<=30 || 165>=beta>=150
114                   f=gegner.pos-ich.pos+5*gegner.ges;
115      elseif 30<beta<50 || 150>beta>130
116                   f=gegner.pos-ich.pos+6*gegner.ges;
117      elseif 50<=beta<=90 || 130>=beta>90
118                  f=gegner.pos-ich.pos+6*gegner.ges;
119      elseif 90<beta
120                   f=gegner.pos-ich.pos+10*gegner.ges;
121      end
122 end
```

Die if-Bedingungen danach enthalten je eine Ausgabe für den Geschwindigkeitsvektor. Bei kleinen Winkeln β soll unsere KillerRaupi relativ direkt auf den Gegner zuhalten und versuchen ihn einzuholen. Bei größeren Winkelabweichungen soll KillerRaupi natürlich nicht einfach nur hinterher fahren, sondern ihm den Weg abschneiden. Dieses Vorhaben wird erreicht, indem der Geschwindigkeitsvektor des Gegners mit verschieden großen Faktoren in die Berechnung des Schubvektors eingeht.

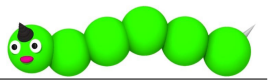
```
123 if ich.radius>1.1*gegner.radius && norm(ich.pos-gegner.pos)<0.3
124      && ich.radius>0.05
125      if 0<beta<15 || 180>beta>175
126      f=gegner.pos-ich.pos+gegner.ges+gegner.ges*1.3;
127      video.text='Angriff2';
128      elseif 15<=beta<=30 || 165>=beta>=150
129                   f=gegner.pos-ich.pos+1.5*gegner.ges;
130      elseif 30<beta<50 || 150>beta>130
131                   f==gegner.pos-ich.pos+3*gegner.ges;
132      elseif 50<=beta<=90 || 130>=beta>90
133                  f=gegner.pos-ich.pos+8*gegner.ges;
134      elseif 90<beta
135                   f==gegner.pos-ich.pos+15*gegner.ges;
136      end
137      video.text='Angriff2';
138 end
```

Der vorangegangene Auszug aus dem Code zeigt noch einen Sonderfall des Angriffs, der dann eingeleitet wird, wenn KillerRaupi nahe am Gegner dran ist. Dann soll KillerRaupi noch stärker auf den Gegner zu halten und vor allem sich ihm noch schneller zu nähern (Faktor in der Berechnung des Schubvektors beträgt 8 und 15). Dementsprechend wird auch als Videotext „Angriff 2“ eingeblendet.

Soweit der Stand der Programmierung von dieser Woche. Grundsätzlich nehmen wir uns für die nächste Zeit vor, noch eine Verteidigungsstrategie zu programmieren, sowie unserer KI einzubläuen, dass sie in keine Minen hinein fährt.
Des Weiteren wollen wir versuchen, Befehle die sich im Code wiederholen zusammen zu fassen, indem diese anfänglich berechnet werden und als Parameter gespeichert werden, um Rechenaufwand zu sparen. Damit kombiniert wollen wir auch versuchen das Tool „flag“ einzubauen, was uns von unserem Professor nahegelegt



wurde einzusetzen.



4.3 3. Projektwoche

Diese Woche ist es uns gelungen den Tankprozess noch weiter zu optimieren. Uns ist aufgefallen, dass der Spaceball manchmal um Tankstellen herum rotiert. Da das Zeit und Sprit kostet haben wir versucht ein besseres Korrekturverfahren einzuführen.
Wie bereits in der vergangenen Woche erläutert, wird ein Winkel bei der Anfahrt zur Tankstelle (alpha1) oder zum Mittelpunkt zwischen 2 Tankstellen (alpha2) berechnet. Wird der Winkel alpha1 nun größer als $10°$, dann muss entsprechend Schub gegeben werden und dabei geht nun der Geschwindigkeitsvektor mit dem Faktor 5 in die Berechnung ein:

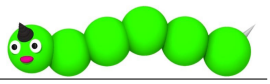
```
1 elseif system~=2 && alpha1>10 && norm(ich.ges)>0.01
2         f=aktuelleTanke.pos-ich.pos-5*ich.ges;
3         video.text='Korrektur 1';
```

Ähnlich verhält es sich mit der Berechnung des Schubvektors, wenn alpha2 zur Ansteuerung des Mittelpunktes zwischen 2 Tankstellen zu stark abweicht. Hier tritt die if-Bedingung erst in Kraft, wenn alpha2 größer als $15°$ wird, d.h. die Toleranz ist etwas größer bei diesem Anfahrverfahren. Auch hier geht der Geschwindigkeitsvektor mit dem Faktor 5 in die Berechnung des Schubvektors ein:

```
1  elseif system==2 && alpha2>15 && norm(ich.ges)>0.01
2         f=((aktuelleTanke.pos+vglTanke.pos)/2)-ich.pos-5*ich.ges;
3         video.text='Korrektur 2';
```

Dieser Faktor sowie die Winkelgrößen wurden innerhalb der Woche durch vielmaliges Probieren und Experimentieren verifiziert und herausgefunden.

Weiterhin haben wir uns mit Stift und Papier überlegt, wie man am günstigsten Minen umfahren kann. Auch das erfolgt vorerst indem man abtastet, ob innerhalb des Bremsweges unseres Spaceballs eine Mine auftaucht. Wenn ja, dann wird Gegenschub radial zur Minenaußenkante gegeben. So würde der Spaceball zwar einen sehr unidealen Haken fahren, aber es würde soweit erst einmal funktionieren.
Des Weiteren mussten wir für diese Woche leider konstatieren, dass unser Spaceball seit Angriff und Verteidigung mit „eingeschaltet" sind, bedeutend häufiger verliert. Daraus ziehen wir den Schluss, dass das Tanken doch eine sehr große Bedeutung besitzt, die in unseren weiteren Überlegungen keinesfalls nach hinten geraten darf.
Außerdem haben wir uns überlegt, dass das derzeitige System der Überlegungen unseres Spaceballs, also dass er sich ein Ziel sucht und dorthin fliegt und eventuell auf Hindernisse stößt, die KI immer nur reagiert, anstatt zu planen. Vielleicht wäre es also denkbar unseren Spaceball so zu programmieren, dass er sich vorausschauend bewegt und nicht erst dann ein Problem erkennt, wenn es direkt vor ihm auftaucht. Dahingehend wäre es beispielsweise denkbar eine Art Gitter auf das Spielfeld zu legen und so von Beginn an zu ertasten, wo genau Tankstellen und Minen liegen und somit eine optimale Route zu finden, die abgefahren werden kann und gar nicht erst Zeit und Sprit mit Korrekturverfahren zu vergeuden. Diese Idee soll in den künftigen Wochen auf jeden Fall weiterentwickelt werden.



4.4 4. Projektwoche

In dieser Woche hat unser Spaceball nun auch grundlegend gelernt sich zu verteidigen, bzw. einfach „wegzulaufen". Das Verteidigungsprinzip beruht derzeit darauf, dass KillerRaupi sich sehr schnell vom Gegner entfernt insofern sie kleiner ist als der Gegner. Das wird als if-Bedingung gesetzt.

```
%% Verteidigung

%Wenn mein radius kleiner ist als der des Gegners, schnell von ihm weg
if ich.radius^2*1.15<=gegner.radius^2>=ich.radius^2*1.2
    f=ich.pos-gegner.pos;
    % wenn ich schneller bin, nicht mehr beschleunigen
    if ich.ges>gegner.ges*1.05
        f=[0 0];
        video.text='gleiten';
    else
        f=ich.pos-gegner.pos;
    end
    video.text='Verteidigung';
end
```

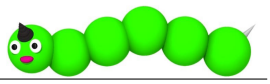
Wenn diese Bedingung erfüllt ist, dann wird Schub genau in Gegenrichtung der Position des Gegners gegeben. Wenn unser Spaceball dann schneller ist als der Gegner schließt sich eine 2. if-Bedingung an), die regelt, dass in diesem Fall dann kein Schub mehr gegeben werden soll, schließlich würde das unnötig Sprit verbrauchen. Ist diese Unterbedingung nicht erfüllt, dann wird weiterhin vom Gegner weg gesteuert.

Der Tankprozess wurde diese Woche quasi unangetastet gelassen, es wurden nur ein paar wenige Veränderungen vorgenommen, die den Rechenaufwand minimieren sollten. Beispielsweise soll nun nicht mehr der Arcuscosinus berechnet werden, sondern bei allen Winkeln nur noch der Cosinus. Weiterhin wurden Wurzeln durch Quadrate ersetzt, sowie überschüssige Klammern weg rationalisiert.

Ähnliches wurde auch im Wandpräventationssystem vorgenommen. Sehr häufig wurde dort der Abstand des Spaceballs von einer Wand berechnet, was 60mal pro Sekunde ein ziemlich großer Aufwand ist. Deshalb werden jetzt einmal zu Beginn die Abstände des Spaceballs in alle Richtungen berechnet und eine Berechnung zum Feststellen der Geschwindigkeit festgelegt und verschiedenen Variablen zugewiesen:

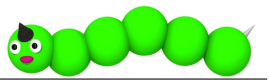
```
ich_abstand_links=ich.pos(1)-ich.radius-0.05;
ich_abstand_rechts=1-ich_abstand_links;
ich_abstand_oben=ich.pos(2)-ich.radius-0.05;
ich_abstand_unten=1-ich_abstand_oben;

ich_schub_max=2*(4e-5/(ich.radius)^2);
ich_schub_halb=2*(2e-5/(ich.radius)^2);
```

Danach schließen sich wie bereits bekannt verschiedene if-Bedingungen an, die regeln wie der Schub eingestellt werden soll um nicht gegen eine Bande zu fahren. Bei den dort enthaltenen Berechnung wurden auch sämtliche Wurzeln durch Quadrieren der Terme eliminiert, damit Rechenleistung gespart werden kann.



```
8  %% Wandpraeventionssystem
9
10 if (ich.ges(1)>0 && ich.ges(1)^2>=ich_schub_max*(ich_abstand_rechts)) || 1-
       ich.pos(1)<ich.radius+0.01
11     % rechts
12     f=[-1 0];
13     video.text='rechte Bande';
14 end
15 if (ich.ges(1)<0 && ich.ges(1)^2>=ich_schub_max*(ich_abstand_links)) ||
       ich.pos(1)<ich.radius+0.01
16     % links
17     f=[1 0];
18     video.text='linke Bande';
19 end
20 if (ich.ges(2)<0 && ich.ges(2)^2>=ich_schub_max*(ich_abstand_oben)) ||
       ich.pos(2)<ich.radius+0.01
21     % oben
22     f=[0 1];
23     video.text='obere Bande';
24 end
25 if (ich.ges(2)>0 && ich.ges(2)^2>=ich_schub_max*(ich_abstand_unten)) || 1-
       ich.pos(2)<ich.radius+0.01
26     % unten
27     f=[0 -1];
28     video.text='untere Bande';
29 end
30 if ((ich.ges(1)>0 && ich.ges(1)^2>=ich_schub_halb*(ich_abstand_rechts))&&(
       ich.ges(2)<0 && ich.ges(2)^2>=ich_schub_halb*(ich_abstand_oben)))
31     % oben rechts
32     f=[-1 1];
33 end
34 if ((ich.ges(1)>0 && ich.ges(1)^2>=ich_schub_halb*(ich_abstand_rechts))&&(
       ich.ges(2)>0 && ich.ges(2)^2>=ich_schub_halb*(ich_abstand_unten)))
35     % unten rechts
36     f=[-1 -1];
37 end
38 if ((ich.ges(1)<0 && ich.ges(1)^2>=ich_schub_halb*(ich_abstand_links))&&(
       ich.ges(2)>0 && ich.ges(2)^2>=ich_schub_halb*(ich_abstand_unten)))
39     % unten links
40     f=[1 -1];
41 end
42 if ((ich.ges(1)<0 && ich.ges(1)^2>=ich_schub_halb*(ich_abstand_links))&&(
       ich.ges(2)<0 && ich.ges(2)^2>=ich_schub_halb*(ich_abstand_oben)))
43     % oben links
44     f=[1 1];
45 end
```

So haben wir erst einmal erreicht, dass der Code insgesamt vorerst übersichtlicher geworden ist und die Rechenleistung soweit es geht gering gehalten wird.



4.5 5. Projektwoche

Diese Woche ist es uns gelungen noch einige Vereinfachungen des Codes in Matlab vorzunehmen. Das Tankverfahren basiert wie bereits beschrieben darauf, dass die Tankstellen in Bezug auf Nähe und Größe miteinander verglichen werden. Das war bisher eine recht große Aneinanderreihung von if-Bedingungen und konnte nun in eine einzelne for-Schleife gepackt werden:

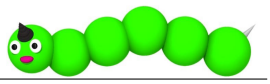
```
%% Mittelpunktverfahren
    vglTanke=spielfeld.tanke(randi([1 10]));
    for i=1:10
        tankei=spielfeld.tanke(i);
        if norm(aktuelleTanke.pos-tankei.pos)<norm(aktuelleTanke.pos-
            vglTanke.pos) && norm(aktuelleTanke.pos-tankei.pos)>0
            vglTanke=spielfeld.tanke(i);
        end
    end
    if  norm(aktuelleTanke.pos-vglTanke.pos)>0 && norm(aktuelleTanke.pos-
        vglTanke.pos)<1.7*ich.radius+aktuelleTanke.radius+vglTanke.radius &&
        vglTanke.radius>0.001
        system=2;
    else
        system=1;
    end
```

Eine for-Schleife hat die Funktion für eine bestimmte Anzahl an Durchgängen den Inhalt der Schleife direkt hintereinander mehrmals zu durchlaufen. In dieser konkreten Anwendung setzt die Schleife für die Variable „i" nacheinander alle Zahlen von 1 bis 10, mit denen eine Tankstelle aus dem Tankstellenvektor „spielfeld.tanke" ausgewählt wird, ein und zieht so jede Tankstelle für den Vergleich heran. Je nachdem ob die if-Bedingungen von der Tankstelle erfüllt werden oder nicht, wird das Mittelpunktverfahren „eingeschalten" (System 2) oder nicht (System 1).

Nach diesem Prinzip wurde auch die Abbruchbedingung zum Tankverfahren, vereinfacht, was nun im Code wir folgt erscheint:

```
%% Abbruchbedingung
abbruchTanke=spielfeld.tanke(randi([1 10]));

for i=1:10
    tankei=spielfeld.tanke(i);
    if tankei.radius>abbruchTanke.radius
        abbruchTanke=spielfeld.tanke(i);
    end
end
if abbruchTanke.radius<0.01
    f=[0 0];
    video.text='Ende';
end
```

Die Funktionsweise dieser Bedingung hat sich hier allerdings nicht verändert. Nach wie vor wird verglichen ob die Tanke noch groß genug zum sinnvollen Tanken ist, wenn alle Tanken leer getankt worden sind, dann wird der Schub abgestellt und es erscheint die Videounterschrift „Ende".

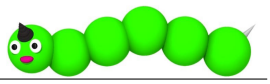


Weiterhin haben die Programmierer begonnen die Idee eines „denkenden" Spaceballs umzusetzen, der zu jeder Zeit im Spiel seinen eigenen perfekten Weg berechnet. Hierfür werden zunächst Vektoren gebildet, die Informationen über die Tankstellen enthalten und diese nach Radius bzw. Abstand der Tankstellen zum Spaceball ordnen.

```
27 %% Sortieren
28 posvec=[spielfeld.tanke.pos];
29 pos1=transpose(posvec(1:2:end));
30 pos2=transpose(posvec(2:2:end));
31
32 A=[pos1,pos2];
33
34 B=sortrows([transpose([spielfeld.tanke.radius]),A],-1);
35
36 for i=1:length(spielfeld.tanke)
37     C(i)=norm(ich.pos-[B(i,2),B(i,3)]);
38 end
39
40 D=sortrows([transpose(C),B(:,1)],1)
```

Zunächst ein Vektor „posvec" definiert, welcher sämtliche Positionen aller Tankstellen enthält. Dieser Vektor wird dann zunächst in 2 Spaltenvektoren aufgeteilt (mittels des Befehls „transpose". Diese Vektoren beinhalten jeweils die x- bzw. die y-Koordinaten der Tankstellen. Die beiden Vektoren werden dann zu einer Matrix zusammengefasst. Der nächste Schritt besteht darin eine Matrix B zu erzeugen, die zum einen neben den Koordinaten der Tankstellen auch deren Radius enthält und das ganze noch nach Größe des Radius geordnet. Im einzelnen fügt sich das zusammen, indem an die Matrix A als zusätzliche Spalte der Spaltenvektor „spielfeld.tanke.radius" angefügt wird, der alle Radien der Tankstellen enthält. Diese Zwischenmatrix wird dann mit dem Befehl „sortrows" nach der Größe des Radius der Tankstellen sortiert und ergibt dann sortiert die Matrix B, welche 3 Spalten und 10 Zeilen umfasst. Die erste Zeile in der Matrix B beinhaltet also: Radius der größten Tankstelle,dazugehörige x-Koordinate,dazugehörige y-Koordinate.
In der darauffolgenden for-Schleife wird der Abstand zwischen jeder Tankstelle und dem Spaceball berechnet und wieder in eine Matrix, diesmal C, gepackt.
Die finale Matrix D enthält nun sortiert den Abstand der Tankstellen und deren Radius, beginnend mit dem kleinsten Abstand und dem zugehörigen Radius.

Das ist natürlich erst einmal der Beginn eines großen Vorhabens unseren Spaceball cleverer als alle anderen zu gestalten, aber zunächst auf jeden Fall ein Schritt auf dem richtigen Weg, den es sich lohnen wird weiter zu gehen.



4.6 6. Projektwoche

Eine recht große Baustelle hatte bis jetzt immer noch das Minensystem dargestellt. Der Spaceball ist, wenn schon nicht direkt in die Mine hineingefahren, immer sehr nahe an die Mine herangefahren, hat dann massiv abgebremst und hat sich dann senkrecht von der Minenoberfläche wieder weggeschoben. Das hat sehr viel Zeit und Sprit vergeudet, weshalb die Idee aufkam die Minen auf dem Weg zu den Tankstellen weiträumig zu umfahren und nicht erst, wenn es fast zu spät ist.
Deshalb haben wir eine function erstellt, mit dem Namen „minensystem", die als Eingangsparameter die Daten von Spielfeld, dem Spaceball und einem Punkt P erhält und einen Schubvektor ausgibt:

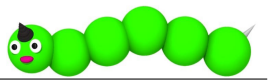
```
1 function [ f ] = minensystem (spielfeld, ich, P)
```

In der folgenden abgebildeten for-Schleife, welche nur in Kraft tritt, wenn die Geschwindgkeit des Spaceballs ungleich Null ist, also er sich bewegt, wird der Spaceball als erstes um eine festgelegte Schrittweite nach vorn geschoben:

```
2 if norm(ich.ges)>0.01
3       for k=0.001:0.05:100
4       neuepos=ich.pos+k*(P-ich.pos);
5       if norm(neuepos-P)<0.05
6           break
7       end
```

Dieses „Vorschieben" erfolgt so lange, bis der Abstand zur aktuellen Tankstelle von dieser neuen Position verschwindend gering ist (if-Bedingung in Zeile 5). Dann wird sie mit dem Befehl „break" abgebrochen. Die verschobene Position wird hierbei als „neuepos" benannt.
Innerhalb dieser ersten for-Schleife ist noch eine zweite for-Schleife enthalten, welche für alle Minen prüft, ob sich der Spaceball an der neuen Position mit einer Mine schneidet bzw. diese berührt und dann mit vielen if-Bedingungen kombiniert eine ideale Route für das Umfahren ermittelt.

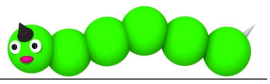
```
8  for i=1:length(spielfeld.mine)
9           if norm(neuepos-spielfeld.mine(i).pos) < ich.radius+spielfeld.mine(i
               ).radius
10              a=ich.ges;
11              b=[-a(2) a(1)];
12              ir=(spielfeld.mine(i).radius+ich.radius+0.02)/norm(b);
13              ad=b*ir;
14              punkt1=(spielfeld.mine(i).pos+ad);
15              punkt2=(spielfeld.mine(i).pos-ad);
16              if norm(ich.pos-punkt1)+norm(P-punkt1) <= norm(ich.pos-punkt2)+
                   norm(P-punkt2)
17                  punkt=punkt1
18              else
19                  punkt=punkt2
20              end
21              cosalpha3=dot(ich.ges, punkt-ich.pos)/(norm(ich.ges)*norm(punkt-
                   ich.pos));
22              if cosalpha3<0.99 && norm(ich.ges)>0.01
23                  f=punkt-ich.pos-8*ich.ges;
24                  video.text='Korrektur 3';
```



```
25                else
26                    if norm(ich.ges)^2>=2*(4e-5/(ich.radius)^2)*norm(ich.pos-P)
27                        f=-ich.ges;
28                    else
29                        f=punkt-ich.pos;
30                        video.text='Versuch';
31                    end
32                end
33            end
34        end
```

Innerhalb der if-Bedingung der Abstand vom Mittelpunkt zwischen neuer Position und einer Mine(i) berechnet. Wenn dieser Abstand kleiner ist als die Summe der Radien von Spaceball und Mine, dann schneiden sich die beiden Objekte. Wenn dieser Fall eintritt, dann wird ein zum Geschwindigkeitsvektor unseres Spaceballs orthogonal stehender Vektor mit der Länge der beiden Radien an die Mine hinan gesetzt. Dieser Punkt liegt also genau so weit von der Mine entfernt, dass der Spaceball gerade so die Mine nicht berührt und folglich ideal ist, da am wenigsten Weg gefahren wird.
In den folgenden beiden Code-Zeilen wird dieser Ideal-Punkt einmal für die rechte und einmal für die linke Seite der Mine berechnet. Dann wird geschaut, welcher Punkt sich eher eignet, das heißt auf welcher Seite es idealer ist die Mine zu umfahren. Um das herauszufinden wird die Gesamtstrecke des Spaceballs von der aktuellen Position über den ausgewählten berechneten Punkt bis zur aktuellen ausgewählten Tankstelle berechnet und dann mit der Route der anderen Seite verglichen. Schlussendlich wird der Punkt ausgewählt, durch den der kürzere Gesamtweg führt. Der Code dazu steckt in einer neuen if-Bedingung.
Analog zum Anfahren der Tankstellen wird im folgenden Teil der Programmierung der Abweichungswinkel berechnet und gegebenenfalls eine Korrektur vorgenommen. Des Weiteren bremst der Spaceball bereits auf dem Weg zu dem Ideal-Punkt ab um dann auch rechtzeitig bei der aktuellen Tankstelle anhalten zu können, sollte das nötig sein.

Zukünftig ist geplant, dass diese function auch das Tankverfahren kombiniert mit beinhaltet.



4.7 7. Projektwoche

Wie bereits in den vergangenen Wochen angekündigt wurde ist es Zeit sich in Sachen Angriff weiter zu entwickeln. Der Code besteht insgesamt aus einer großen if-Bedingung. Zu Beginn dieser if-Bedingung wird grundsätzlich erst einmal festgelegt, wann angegriffen werden soll.

```
1  if ich.radius>1.15*gegner.radius...
2          && ich.radius>0.035...
3          && norm(ich.pos-gegner.pos)<0.6...
4          && norm(ich.pos-aktuelleTanke.pos)>0.1...
5          ||ich.radius>1.05*gegner.radius...
6          && ich.radius>0.035...
7          && norm(ich.pos-gegner.pos)<0.1...
8          && norm(ich.pos-aktuelleTanke.pos)>0.1
```

Auf jeden Fall nur dann, wenn der Radius unseres Spaceballs größer ist als der des Gegners und unser Radius eine Mindestgröße überschreitet. Weiterhin wollen wir auch nicht besonders weit weg vom Gegner sein und nicht direkt neben einer Tankstelle stehen. Diese Bedingungen folgen dann noch einmal mit „oder (||)" angehangen ein wenig anders: Wenn der Spaceball näher am Gegner dran ist aber nicht so viel größer als dieser, wird trotzdem angegriffen.

Im weiteren Code, immer noch in der gleichen if-Bedingung, schließen sich einige Berechnungen über Ort und Geschwindigkeit der beiden Spaceballs an, sowie die Winkel zwischen den entsprechenden Vektoren.

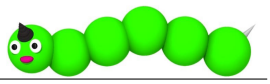
```
9      e=gegner.pos-ich.pos;
10     f=(gegner.ges)-(ich.ges);
11
12     R12=ich.radius+gegner.radius;
13
14     alpha=dot(f,f);
15     beta=dot(e,f);
16     gamma=dot(e,e)-R12^2;
17
18     delta=beta^2-alpha*gamma;
```

Anhand dieser Art Kollisionsanalyse werden nun Zeitpunkte berechnet:

```
19 if delta>0
20         la1=(-beta+sqrt(delta))/alpha
21         la2=(-beta-sqrt(delta))/alpha
22
23         [la]=sort([la1 la2]);
```

Zu diesen Zeitpunkte la1 und la2 würden Kollisionen stattfinden und diese werden nun in einen gemeinsamen Vektor la gesteckt, welcher dann noch nach Größe (kleinster zuerst) sortiert wird.

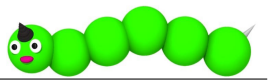
```
24 if la(1)<1
25             zl=ich.pos+la(1)*ich.ges
26
27             if zl(1)<1.05 && zl(1)>-0.05 && zl(2)<1.05 && zl(2)>-0.05
28                 f=zl
```



```
29                end
30
31            end
32            video.text='Angriff'
```

Wenn die Dauer bis zum Angriff kleiner als 1 ist, dann wird der Punkt berechnet, an dem die Spaceballs aufeinander treffen. Wenn dieser Vektor noch einmal die Bedingung erfüllt im Spielfeld zu liegen, dann wird auf diesen Punkt zugesteuert und es erfolgt der Angriff.

Der Angriff soll in nächster Zeit noch ein wenig komplexer gestaltet werden: Es soll nur dann angegriffen werden, wenn auf dem Weg keine Mine liegt (damit sollen unnötige Gefahren umgangen werden) und es sollen auch noch verschiedene Sonderfälle betrachtet werden, bspw. wie unser Spaceball handeln soll, wenn die Spaceballs genau hintereinander fahren oder sich genau entgegengesetzt bewegen.



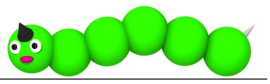
4.8 8. Projektwoche

Schon seit längerer Zeit hatten wir uns vorgenommen, um Rechenaufwand zu sparen und uns selbst die Verkettung der einzelnen Funktionen unserer KI (Tanken, Minen, Wand, Angriff) zu vereinfachen, mit sogenannten „flags" zu arbeiten. Diese haben wir nun diese Woche zuerst versuchsweise in das Wandpräventationssystem eingebaut. Die flags dienen dem gleichzeitigen Erkennen von mehreren Hindernissen. Fuhr unsere alte KI beispielsweise auf eine Mine und eine Wand zu, so konnte schrittweise entweder die Mine oder die Wand betrachtet werden. Das führte dazu, dass in der Animation des Spiels der Schubvektor sehr unschön geflackert hat und nahezu jedes Mal der Spaceball entweder Mine oder Wand doch noch touchiert hat. Statt nun direkt nach dem Erkennen der Gefahr einen Schubvektor f herauszugeben wird erst einmal ganz bürokratisch vom System registriert aber keine überstürzten Handlungen vollzogen. Einen flag zu setzen bedeutet also konkret nichts anderes als eine Gefahrenmeldung abzugeben. Am Ende der kraft-Datei kann dann der beste und vorteilhafteste Schubvektor unter Berücksichtigung aller Hindernisse berechnet werden.
Derzeit ist nur das Antiwand-System mit flags programmiert, sobald das Minensystem steht soll dieses natürlich auch alsbald mit flags ausgerüstet werden.
Zur genauen Erläuterung dient also hier das Antiwandsystem. Als erstes werden dort alle Variablen auf inaktiv („0") gesetzt:

```
1 flag_bande_rechts=0;
2 flag_bande_links=0;
3 flag_bande_oben=0;
4 flag_bande_unten=0;
5 flag_bande_obenlinks=0;
6 flag_bande_obenrechts=0;
7 flag_bande_untenrechts=0;
8 flag_bande_untenlinks=0;
```

Diese Gefahrenmelder werden dann in die if-Bedingungen des Wandpräventationssystems mit eingebunden und wenn eine der Bedingungen zutrifft, wird diese flag aktiviert („1"), d.h. die Meldung herausgegeben. Ein Auszug des Codes ist in der nächsten Box zu sehen.

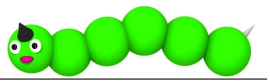
```
9  if (ich.ges(1)>0 && ich.ges(1)^2>=ich_bes_max*(ich_abstand_rechts)) || 1-
       ich.pos(1)<ich.radius+0.01
10     % rechts
11     flag_bande_rechts=1;
12     video.text='rechte Bande';
13 end
14
15 if (ich.ges(1)<0 && ich.ges(1)^2>=ich_bes_max*(ich_abstand_links)) || ich.pos
       (1)<ich.radius+0.01
16     % links
17     flag_bande_links=1;
18     video.text='linke Bande';
19 end
20
21 if ((ich.ges(1)>0 && ich.ges(1)^2>=ich_bes_halb*(ich_abstand_rechts))&&(
       ich.ges(2)<0 && ich.ges(2)^2>=ich_bes_halb*(ich_abstand_oben)))
22     % oben rechts
```



```
23      flag_bande_obenrechts=1;
24  end
25
26  if ((ich.ges(1)>0 && ich.ges(1)^2>=ich_bes_halb*(ich_abstand_rechts))&&(
        ich.ges(2)>0 && ich.ges(2)^2>=ich_bes_halb*(ich_abstand_unten)))
27      % unten rechts
28      flag_bande_untenrechts=1;
29  end
```

Schlussendlich werden die flags noch ausgewertet, das heißt wenn ein oder mehrere flag aktiviert sind, wird hier noch bestimmt, wie der Schubvektor aussehen soll. Das sind im Folgenden natürlich erst einmal noch simple Zuweisungen, die aber bald durch Berechnungen ersetzt werden sollen.

```
30  if(flag_bande_rechts) f=[-1 0]; end
31  if(flag_bande_links) f=[1 0]; end
32  if(flag_bande_oben) f=[0 1]; end
33  if(flag_bande_unten) f=[0 -1]; end
34  if(flag_bande_obenrechts) f=[-1 1]; end
35  if(flag_bande_untenrechts) f=[-1 -1]; end
36  if(flag_bande_untenlinks) f=[1 -1]; end
37  if(flag_bande_obenlinks) f=[1 1]; end
```



4.9 9. und 10. Projektwoche

Unsere KillerRaupi hatte es in den vergangenen Wochen perfektioniert zu tanken, weshalb sie sich in vielen Simulationen als extrem tank-wütig herausgestellt hat. Das führte dazu, dass KillerRaupi häufig nicht zu Ende getankt hat, sondern, sobald eine nahe Tankstelle größer war als die derzeitige, zu dieser neuen Tankstelle sofort hingefahren ist. Da das extrem ineffektiv und gefährlich ist, mussten die Programmierer hier noch einmal ran. Da sich diese Arbeit über 2 Wochen erstreckt hat, beschloss die Dokumentarin diese beiden Projektwochen zusammenzufassen.

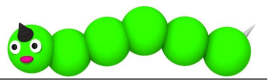
```
1 for i=1:length(spielfeld.tanke)
2       if spielfeld.tanke(i).radius>0.015 && (norm(ich.pos-spielfeld.tanke(i)
            .pos)<(ich.radius+spielfeld.tanke(i).radius))
3           f=-ich.ges;
4           if norm(ich.ges)<0.005 || norm(ich.pos-spielfeld.tanke(i).pos)<0.01
5               f=[0 0];
6           end
7           video.text='Stop!2';
8   end
9 end
```

Diese for-Schleife prüft zuerst, ob unser Spaceball eine andere Tankstelle schneidet. Wenn das der Fall ist und die Tankstelle einen Radius größer als 0.015 hat, dann wird gebremst (1. if-Bedingung der Schleife). Wenn wir uns der Tankstelle mit sehr kleiner Geschwindigkeit nähern oder wir sehr nahe an der Tankstelle dran sind, dann wird der Schub abgestellt.
Ein gewisses Problem stellt allerdings dar, dass diese Bedingung das Mittelpunktverfahren zum Tanken aushebelt. Eine mögliche Lösung wäre es, diese Bedingung nur einzufügen, wenn der normale Tankmodus aktiviert ist.

Weiterhin ist uns bei den viele Simulationen aufgefallen, dass unser Spaceball (der Rote) immer verliert, wenn der Gegner und KillerRaupi gleich viel Sprit haben, da es ja kein Unentschieden gibt. Das brachte uns auf die Idee noch einen zusätzlichen Einschub in unsere bestehende Abbruch-Bedingung einzufügen, die unseren Spritverbrauch auf ein Minimum reduziert:

```
10 elseif abbruchTanke.radius<0.01 && ich.radius<=gegner.radius && norm(ich.ges)
       <0.02
11      f=aktuelleTanke.pos-ich.pos;
12 elseif abbruchTanke.radius<0.01 && ich.radius<=gegner.radius && norm(ich.ges)
       >=0.02
13      f=[0 0];
```

Wenn unser Radius kleiner gleich der des Gegners ist, dann geben wir Schub in Richtung einer der übrig gebliebenen Tankstellen, allerdings nur sehr geringen, damit wir maximal eine Geschwindigkeit von 0.02 erreichen. Wenn diese Geschwindigkeit erreicht ist, dann stellen wir den Schub auf Null und segeln somit energiesparend und ökonomisch auf unsere ausgewählte Tankstelle zu.
Eine geplante Verbesserung ist hierbei allerdings noch zu prüfen, ob der direkte Weg nicht durch eine Mine versperrt ist. Sollte dies so sein, müsste eine andere Tankstelle ausgewählt werden.



Um unsere Gewinnchancen noch weiter zu steigern wurde außerdem eingeführt, dass KillerRaupi zu tanken aufhört, wenn wir mehr Treibstoff gebunkert haben, als der Gegner noch erreichen kann:

```
14 if ich.radius > gegner.radius ...
15         && ich.radius^2 > gegner.radius^2 ...
16         + sum([spielfeld.tanke(1:length(spielfeld.tanke)).radius].^2)
17     f=-ich.ges;
18     if norm(ich.ges)<0.001
19         f=[0 0];
20     end
21     video.text='Treibstoff';
22 end
```

Dazu wird berechnet, ob die Fläche unseres Spaceballs größer ist, als die Summe aller übrigen Tankstellen und die Fläche des Gegners. Wenn das der Fall ist, dann hören wir auf zu tanken, bremsen ab und sitzen den Rest des Spiels quasi aus. Das ist vielleicht langweilig, aber leider sehr effektiv und sicher.

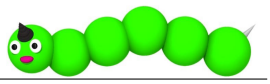
In den letzten Tagen haben wir auch noch gelernt, dass unser Minenabwehrsystem und das Antiwandsystem einzeln zwar sehr gut funktioniert, wenn jedoch der Spaceball auf Mine und Wand zufährt, dann kommen sich die Abwehrsysteme gegenseitig in die Quere und der Spaceball scheitert glorreich. Deshalb haben wir noch eine if-Bedingung in das kraft-file ergänzt, die genau diese Gefahr abtastet.

```
23 if doppelteGefahr(spielfeld,ich,flag_bande_rechts)==1 || doppelteGefahr(
       spielfeld,ich,flag_bande_links)==1 ...
24         || doppelteGefahr(spielfeld,ich,flag_bande_oben)==1 || doppelteGefahr
               (spielfeld,ich,flag_bande_unten)==1 ...
25         || doppelteGefahr(spielfeld,ich,flag_bande_obenrechts)==1 ||
               doppelteGefahr(spielfeld,ich,flag_bande_untenrechts)==1 ...
26         || doppelteGefahr(spielfeld,ich,flag_bande_untenlinks)==1 ||
               doppelteGefahr(spielfeld,ich,flag_bande_obenlinks)==1
27     f=-ich.ges
28     video.text='Doppelt';
29 end
```

Hierbei wird geschaut, ob die flags, also die Warnungen für Mine und Wand aktiviert ist, und die entsprechende Maßnahme eingeleitet, was hier konkret bedeutet: Umkehrschub und stark abbremsen, damit man nicht in die Gefahr hineingerät.

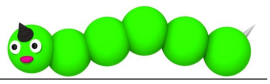
Das System hierzu selbst ist noch einmal extern in der folgenden Funktion formuliert:

```
30 function [ gefahr ] = doppelteGefahr(spielfeld,ich,flag)
31
32 for i=1:length(spielfeld.mine)
33     minei=spielfeld.mine(i);
34     winkelzupunkt=atan((minei.radius+ich.radius+0.01)/norm(ich.pos-minei.pos)
           );
35     winkeligeszumine=acos(dot(ich.ges, minei.pos-ich.pos)/(norm(ich.ges)*norm
           (minei.pos-ich.pos)));
36     if flag && (norm(ich.ges)^2>=2*(4e-5/(ich.radius)^2)*(norm(minei.pos-
           ich.pos)-ich.radius-minei.radius-0.01) ...
37             && winkeligeszumine-winkelzupunkt<0) || flag && (norm(minei.pos-
                   ich.pos)-minei.radius-ich.radius)<0.009
```



```
38          gefahr=1;
39      else
40          gefahr=0;
41      end
42  end
43
44  end
```

Hier werden zuerst die Winkel zwischen uns und den verschiedenen Minen berechnet und somit geschaut, ob sich unser Spaceball auf eine Mine zubewegt. Wenn eine flag aus dem Antiwandsystem aktiviert ist und KillerRaupi sich tatsächlich auf eine Mine zubewegt und sich sehr kurz vor ihr befindet, oder der Bremsweg zur Mine zu knapp wird, dann wird die Gefahr eingeschaltet, also auf 1 gesetzt.
Seit wir dieses System eingegliedert haben ist es zwar scheinbar noch nie angesprungen, aber dafür ist unser Spaceball auch noch nie wieder in so eine Doppelgefahr hineingefahren. Das klingt unlogisch, ist es auch.



4.10 11. Projektwoche

Wir sind doch wieder in so eine Doppelgefahr hineingefahren. Da somit die Funktionalität dieser function stark anzuzweifeln war, haben wir beschlossen sie im m-File erst einmal wieder auszukommentieren. Denn wir mussten auch leider feststellen, dass diese function - und damit das gesamte kraft-file - versagt, sobald die Minenanzahl auf 0 gestellt wird und das zwingt uns ebenfalls Abstand von der Doppeltengefahr-Funktion zu nehmen. Schade eigentlich - cool war sie allemal.

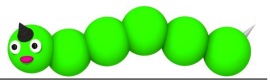
Da wir irgendwie bei der Minenabwehr ein wenig stagnieren, weil uns auf unerklärliche Weise ein besserer Denkansatz fehlt, haben wir uns in der vergangenen Woche damit beschäftigt das Tanken weiter zu verbessern (ja, Verbesserungen sind immer möglich). Hierfür wurde ein neuer Wert in die Auswahl unserer potentiellen Tankstellen mit eingefügt, der den Abstand des Gegners zur gerade betrachteten Tankstelle beinhaltet. Dafür berechnet man den Abstand zwischen den beiden besagten Objekten und zieht diesen vom „Tankstellenbewertungsergebnis" noch ab. Je kleiner damit das „Tankstellenbewertungsergebnis" ist, desto geeigneter ist diese Tankstelle. Das wird besonders dann relevant, wenn wir uns in der Situation befinden, dass zwei Tankstellen gleich weit entfernt von uns sind, der Gegner aber an einer Tankstelle bedeutend näher dran ist als wir. Dann kämen wir dem Gegner nicht in die Quere, was bei einem symmetrischen Spielfeld leider sehr oft passieren kann. Der errechnete Abstandswert wird immer relativ klein sein, schließlich wollen wir nicht, dass die Position des Gegners unsere Tankstrategie soooooo entscheidend beeinflusst, sondern sie soll ja wirklich nur dann in Kraft treten, wenn wir ungefähr gleich weit weg von 2 Tankstellen sind und dann den entscheidenden Unterschied ausmachen. Der Code dazu ist im folgenden Kästchen abgebildet.

```
1  for i=1:length(spielfeld.tanke)
2      tankei=spielfeld.tanke(i);
3      if tankei.radius>0.005 && norm(ich.pos-tankei.pos)/2-norm(gegner.pos-
           tankei.pos)/5-10*tankei.radius ...
4              <= norm(ich.pos-aktuelleTanke.pos)/2-norm(gegner.pos-
                  aktuelleTanke.pos)/5-10*aktuelleTanke.radius
```

Danach folgt in der function aktuelleTankebestimmen noch eine Betrachtung, ob der Gegner die eben ausgewählte Tankstelle bereits schneidet und ob der Gegner größer ist als wir. Wenn der Gegner die Tankstelle nicht schneidet oder sie schneidet aber kleiner ist, dann okkupieren wir die Tankstelle trotzdem. Wenn der Gegner die Tankstelle jedoch schneidet UND größer ist, dann lassen wir die Finger davon.

```
5      if norm(tankei.pos-gegner.pos)>tankei.radius+gegner.radius
6          aktuelleTanke=spielfeld.tanke(i);
7
8          elseif norm(tankei.pos-gegner.pos)<tankei.radius+gegner.radius
9              if gegner.radius<ich.radius
10                 aktuelleTanke=spielfeld.tanke(i);
11             end
12         end
13     end
```

Dieser Gedanke soll in nächster Zeit noch dahingehend ausgebaut werden, dass

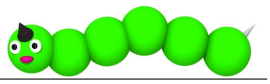


betrachtet wird, ob der Gegner die Tankstelle vor uns erreichen könnte und ob er größer oder kleiner als unser Spaceball ist. Dafür könnte man auch die von uns erstellte Kollisionsanalyse mit einbinden, um zu bestimmen ob man die Tankstelle annektieren sollte oder bessert nicht, weil der Gegner zum Beispiel bis zu unserem Eintreffen größer als wir sein könnte.

Weiterhin haben wir uns noch eine andere kleine Raffinesse ausgedacht. Da man relativ begründet annehmen kann, dass jeder Gegner versuchen wird alle Tankstellen aufzutanken, sind wir auf die Idee gekommen genau diese „Schwäche" auszunutzen. Wenn wir also bereits mehr Sprit als der Gegner getankt haben und nur noch eine große Tankstelle übrig ist, dann platzieren/ legen wir uns in geringem Abstand (0.01) davor auf die Lauer und warten ganz ruhig bis der Gegner vorbeikommt um dann anzugreifen. So vergeuden wir keinen Sprit um dem Gegner auf dem Feld hinterherzuhetzen und ihn, da er wendiger ist, wahrscheinlich nicht einzufangen.

```
14 if minennummer(spielfeld,ich,ich.pos,aktuelleTanke.pos)==0 &&
       aktuelleTanke.radius>0.01
15          f=aktuelleTanke.pos-ich.pos;
16          if norm(ich.ges)^2>2*(4e-5/(ich.radius)^2)*(norm(ich.pos-
                aktuelleTanke.pos)-0.01-aktuelleTanke.radius-ich.radius)
17              f=-ich.ges;
18              if norm(ich.ges)<0.0005
19                  f=[0 0];
20              end
21          end
22      end
```

Diese Verfahren wird allerdings nur dann angewendet, wenn der Weg zu der besagten Tankstelle nicht von einer Mine versperrt ist an der wir uns wenn es ganz blöd läuft sogar noch verheddern könnten und die sicheren Siegespunkte noch verloren gehen.



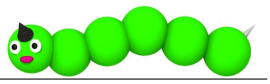
4.11 12. Projektwoche

In den letzten Wochen ist uns in mehreren Simulationen noch aufgefallen, dass unser Minenabwehrsystem unter anderem genau dann versagt, wenn 2 Minen relativ dicht nebeneinander liegen, zumindest so dicht, dass unser Spaceball nicht durch den Abstand dazwischen hindurchpasst, dann hängt sich unsere KillerRaupi fest. Dafür haben wir nun nach einer Lösung gesucht und diese auch gefunden.
Wir ersetzen diese zwei kuschelnden Minen durch eine große imaginäre. Der Abstand zwischen den beiden Minen entspricht dabei dem Durchmesser unserer imaginären Mine. Im Code ist dieses System in zwei for-Schleifen mit einer if-Bedingung verankert. Hierbei wird dann erst die Position der Mine festgelegt und anschließend ein entsprechender Schub definiert um die Minen erfolgreich zu umfahren:

```
1  for h=1:length(spielfeld.mine)
2      for g=1:length(spielfeld.mine)
3          if g>h && norm(spielfeld.mine(h).pos-spielfeld.mine(g).pos)<
               spielfeld.mine(h).radius+spielfeld.mine(g).radius+2*ich.radius+0
               .01
4              zusatzmine.pos=(spielfeld.mine(h).pos+spielfeld.mine(g).pos)/2;
5              zusatzmine.radius=(norm(spielfeld.mine(h).pos-spielfeld.mine(g)
                   .pos)/2)-spielfeld.mine(h).radius-spielfeld.mine(g).radius;
6
7
8              winkelzupunkt=atan((zusatzmine.radius+ich.radius+0.01)/norm(
                   ich.pos-zusatzmine.pos));
9              winkeligeszumine=acos(dot(ich.ges, zusatzmine.pos-ich.pos)/(norm(
                   ich.ges)*norm(zusatzmine.pos-ich.pos)));
10
11             if norm(ich.ges)^2>=2*(4e-5/(ich.radius)^2)*(norm(zusatzmine.pos-
                   ich.pos)-ich.radius-zusatzmine.radius-0.01) ...
12                 && winkeligeszumine-winkelzupunkt<0
13                 f=-ich.ges;
14             elseif(norm(zusatzmine.pos-ich.pos)-zusatzmine.radius-ich.radius)
                   <0.009
15                 f=ich.pos-zusatzmine.pos;
16             end
17         end
18     end
19 end
```

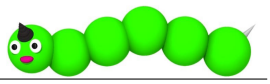
Bedauerlicher Weise ist unser System wohl noch nicht so richtig perfekt, denn an den Ecken, also die Stellen, wo sich die „drei" Minen gerade berühren, bleibt unser Spaceball doch manches Mal hängen. Vielleicht ersetzt man den bisherigen Durchmesser der imaginären Mine durch den Abstand zwischen den beiden Minen plus deren Durchmesser, damit die imaginäre Mine alles umfasst und es keine Ecken mehr gibt.

Weiterhin haben wir unser Treibstoff-Spar-System noch weiterentwickelt. Die Überlegung hinter diesem System war, dass unser Spaceball sobald er mehr Treibstoff besitzt als der Gegner noch erreichen kann, einfach stehen bleibt. Das haben wir noch ein wenig modifiziert. Wenn der Gegner sich unserem Spaceball auf ein geringes Maß (Abstand kleiner als 0.2) nähert, dann gibt unser Spaceball, der ja größer ist, Schub in



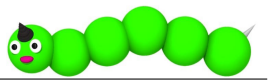
dessen Richtung. Wenn wir Glück und der Gegner ein schlechtes Verteidigungssystem haben, dann ist es uns möglich ihn zu erwischen. Wenn der Gegner sich wieder entfernt, so wird der Vorgang abgebrochen und Sprit gespart.

```
1  if norm(ich.pos-gegner.pos)<0.2
2          f=gegner.pos-ich.pos-ich.ges;
3      end
```



4.12 13. Projektwoche

In dieser Projektwoche ist nichts direkt Neues mehr in Sachen Programmierung entstanden. Alte Systeme wurden noch ein wenig angepasst, vieles zusammengefügt, einiges aussortiert und das A und O: sehr häufig simuliert. Die Hauptaufgabe unserer Programmierer bestand aufgrund des näher rückenden Abgabetermines eher darin, alle Teilfunktionen und -stücke des kraft-Datei ineinander zu packen und aufeinander abzustimmen. Eine relativ ausführliche Beschreibung der einzelnen Funktionen im Codes ist im nachfolgenden Kapitel „endgültiger KI-Code" zu finden.



4.13 Endgültiger KI-Code

Der Code der unsre KillerRaupi als Spaceball fahren lässt wurde mit MATLAB Simulink 2013 erstellt. Das so entstandene „kraft-file" hat zur Aufgabe je nach Situation einen Schubvektor f auszugeben, der an ein übergeordnetes m-file übermittelt wird und unseren Spaceball in eine bestimmte Richtung fahren lässt. 60 mal pro Sekunde wird dieses kraft-file durchlaufen und somit auch 60 mal ein Schubvektor berechnet.
Unser endgültiger KI-Code im m-file umfasst grundsätzlich 439 Zeilen.
Zu Beginn müssen erst einmal diverse Definitionen erwähnt werden, bezüglich Spielfeld und der Farbgebung unserer Spaceballs.

```
1  function [ f, video ] = kraft( spielfeld, farbe )
2
3  if strcmp (farbe, 'rot')
4      ich=spielfeld.rot;
5      gegner=spielfeld.blau;
6  else
7      ich=spielfeld.blau;
8      gegner=spielfeld.rot;
9  end
10
11 f=[0 0];
```

Dann folgt eine erste kurze Programmierung unsererseits, die herausfindet wie viele Minen sich auf dem Spielfeld befinden, da diese Anzahl variieren kann. Weiterhin wird den Minen ein Geschwindigkeitsvektor der Größe Null zugewiesen, da wird diese Angabe später für die Kollisionsanalyse benötigen.

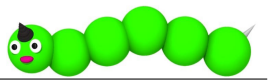
```
12 for u=1:length(spielfeld.mine)
13     spielfeld.mine(u).ges=0;
14 end
```

Danach erscheinen wieder einige notwendige Festsetzungen zum Thema Videoeinblendung und Texteinblendung, sowie eine Aktivierung des Tankmodus, der nahezu immer aktiviert sein sollte.

```
15 video.index=0;
16 video.text='';
17 tanken=1;
```

Im Folgenden erscheint mithilfe der Kollisionsanalyse (Erklärung dieser Funktion erfolgt im weiteren Verlauf der Zusammenfassung) eine Berechnung, die ermittelt ob sich zwischen unserer Position und der des Gegner eine Mine befindet. Dementsprechend wird eine 1 ausgegeben für: eine Mine ist im Weg; und eine 0 für: es ist keine Mine im Weg.

```
18 for t=1:length(spielfeld.mine)
19     m=ka(ich, spielfeld.mine(t));
20     if isempty(m.la1)==0 && norm(ich.pos-m.kreis1pos1)<norm(ich.pos-
           gegner.pos)
21         mineImWeg=1;
```



```
22      else
23          mineImWeg=0;
24      end
25  end
```

Nach diesem kurzen Einschub zum Gegner folgt die Aufzählung aller möglichen flags, die hierbei alle erst einmal auf 0, also inaktiv, gesetzt werden.

```
26  flag_bande_rechts=0;
27  flag_bande_links=0;
28  flag_bande_oben=0;
29  flag_bande_unten=0;
30  flag_bande_obenlinks=0;
31  flag_bande_obenrechts=0;
32  flag_bande_untenrechts=0;
33  flag_bande_untenlinks=0;
```

Weiterhin folgen allgemeine Berechnungen auf die an anderen Stellen im m-file noch zugegriffen werden muss. Hier werden die Abstände unseres Spaceballs zur Bande berechnet, sowie eine Maximalgeschwindigkeit und eine gemächlichere Geschwindigkeit definiert.

```
34  ich_abstand_links=ich.pos(1)-ich.radius-0.01;
35  ich_abstand_rechts=1-(ich.pos(1)+ich.radius+0.01);
36  ich_abstand_oben=ich.pos(2)-ich.radius-0.01;
37  ich_abstand_unten=1-(ich.pos(2)+ich.radius+0.01);
38
39  ich_bes_max=2*(4e-5/(ich.radius)^2);
40  ich_bes_halb=2*(2e-5/(ich.radius)^2);
```

Jetzt wird es zum ersten Mal so richtig interessant, denn es folgt unser **Tanksystem**. Zuerst wird nach bestimmten Kriterien eine aktuelle Tankstelle ausgewählt, welche in der function „aktuelleTankeBestimmen" festgesetzt sind (darauf wird gegen Ende des m-files noch genauer eingegangen).

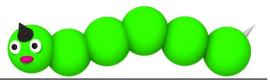
```
41  aktuelleTanke=aktuelleTankeBestimmen(spielfeld, ich, gegner);
```

Wenn eine Tankstelle zu klein geworden ist, dann muss einen neue Tankstelle bestimmt werden.

```
42  if(aktuelleTanke.radius<0.005)
43      aktuelleTanke=aktuelleTankeBestimmen(spielfeld, ich, gegner);
44  end
```

Danach schließt sich das sogenannte Mittelpunktverfahren an, auf das wir schon recht stolz sind, da wir uns damit sehr viel von dem auf dem Spielfeld verfügbaren Sprit sichern können. Grundsätzlich wird dieses Verfahren nur dann durchlaufen, wenn unser Tankmodus angeschalten ist:

```
45  if(tanken==1)
```



Danach wird per Zufall eine der vorhandenen Tankstellen ausgewählt und mit allen anderen Tankstellen verglichen in Bezug auf Größe und Nähe zu der aktuellen Tankstelle, die wir ausgewählt haben. Denn wir wollen eine neue Vergleichstankstelle auswählen, die möglichst nahe an der derzeitig ausgewählten daran liegen soll und damit Potential für das Mittelpunktverfahren aufweisen würde.

```
46 vglTanke=spielfeld.tanke(randi([1 length(spielfeld.tanke)]));
47     for i=1:length(spielfeld.tanke)
48         tankei=spielfeld.tanke(i);
49         if norm(aktuelleTanke.pos-tankei.pos)-tankei.radius<norm(
               aktuelleTanke.pos-vglTanke.pos)-vglTanke.radius ...
50                 && norm(aktuelleTanke.pos-tankei.pos)>0 && tankei.radius>0
51             vglTanke=spielfeld.tanke(i);
52         end
53     end
```

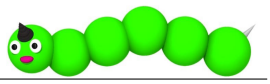
Nach dieser for-Schleife schließt sich die eigentliche Abfrage zum Mittelpunktverfahren an. Wenn die aktuelle Tankstelle und die Vergleichstankstelle so nah beieinander liegen dass sie überhaupt das Potential besitzen für ein Mittelpunkttanken in Frage zu kommen, dann wird hier abgefragt, ob der Radius unseres Spaceballs groß genug ist um die Lücke zwischen den beiden Tankstellen zu schließen (sogar noch mit einem Sicherheitsfaktor von 1,7). Dann wird unser System auf die Schaltung 2 gestellt. Wenn das nicht der Fall ist, dann wird auf 1 geschaltet.

```
54 if  norm(aktuelleTanke.pos-vglTanke.pos)>0 ...
55             && norm(aktuelleTanke.pos-vglTanke.pos)<1.7*ich.radius+
                   aktuelleTanke.radius+vglTanke.radius ...
56             && vglTanke.radius>0.001
57         system=2;
58     else
59         system=1;
60     end
```

Im weiteren Verlauf der if-Bedingung unseres Tankmodus werden dann einige Berechnungen angestellt. Als erstes wird der Cosinus des Winkel alpha1, welcher den Winkel zwischen unserem Geschwindigkeitsvektor und dem Mittelpunkt unserer ausgewählten Tankstelle darstellt, berechnet.
Der Cosinus des Winkels alpha2 gehört zum Mittelpunktverfahren und beschreibt dementsprechend den Winkel zwischen unserem Geschwindigkeitsvektor und dem Vektor zwischen unserer Position und dem Mittelpunkt der beiden ausgewählten Tankstellen. Auch dieser wird berechnet. Um Rechenleistung zu sparen wird hierbei immer nur der Cosinus des zugehörigen Winkels berechnet, nicht der Winkel selbst.

```
61 cosalpha1=dot(ich.ges, aktuelleTanke.pos-ich.pos)/(norm(ich.ges)*norm(
       aktuelleTanke.pos-ich.pos));
62     if system==2
63         cosalpha2=dot(ich.ges,(aktuelleTanke.pos+vglTanke.pos)/2-ich.pos)/(
               norm(ich.ges)*norm((aktuelleTanke.pos+vglTanke.pos)/2-ich.pos));
64     end
```

Auf Basis dieser Winkel kann nun die Steuerung unseres Spaceballs bei der Anfahrt der ausgewählten Tankstellen erfolgen.
Die erste if-Bedingung beinhaltet das Bremsen, wenn unser Spaceball die Tankstelle



berührt. Wenn unsere Geschwindigkeit sehr gering ist, dann wird kein Schubvektor mehr ausgegeben um das Flackern zu vermeiden.

```
65  if (norm(ich.pos-aktuelleTanke.pos)<(ich.radius+aktuelleTanke.radius)) &&
        system==1 && aktuelleTanke.radius>0.02
66          f=-ich.ges;
67          if norm(ich.ges)<0.005 || norm(ich.pos-aktuelleTanke.pos)<0.01
68              f=[0 0];
69          end
70          video.text='Stop!';
```

In der nächsten if-Bedingung wird ein Schubvektor definiert, der dann anzuwenden ist, wenn der Winkel zwischen Geschwindigkeit und Tankstelle größer als $8,1°$ $(arccos(0,99))$ ist, d.h. eine Korrektur vorgenommen werden muss.

```
71 elseif system==1 && cosalpha1<0.99 && norm(ich.ges)>0.02
72          f=aktuelleTanke.pos-ich.pos-5*ich.ges;
73          video.text='Korrektur 1';
```

Danach folgt die entsprechende Korrektur für das Mittelpunktanfahren, hier wird ebenfalls eine Abweichung des Winkel alpha2 von $8,1°$ $(arccos(0,99))$ als Kriterium gestellt.

```
74 elseif system==2 && cosalpha2<0.99 && norm(ich.ges)>0.02
75          f=((aktuelleTanke.pos+vglTanke.pos)/2)-ich.pos-5*ich.ges;
76          video.text='Korrektur 2';
```

Wenn keine Korrektur nötig ist, kann der Mittelpunkt zwischen den Tankstellen normal angefahren werden, was in den nächsten Zeilen codiert ist:

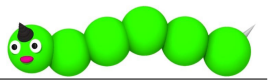
```
77 elseif system==2 && aktuelleTanke.radius>0.01
78          f=((aktuelleTanke.pos+vglTanke.pos)/2)-ich.pos;
79          if norm((aktuelleTanke.pos+vglTanke.pos)/2-ich.pos)<0.02 && norm(
                ich.ges)<0.005
80              f=[0 0];
81          end
82          video.text='Mittelpunkt';
```

Ebenso gibt es eine normale Ansteuerung der aktuellen Tankstelle:

```
83      else
84          f=aktuelleTanke.pos-ich.pos;
85          video.text='Zielen!';
86      end
87  end
```

Danach schließt sich eine for-Schleife an, die verhindert, dass unser Spaceball eine Tankstelle nicht vollständig leert, weil sie ihm zu klein geworden ist und er eine attraktivere gefunden hat.

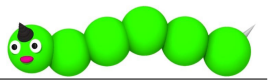
```
88  for i=1:length(spielfeld.tanke)
89      if spielfeld.tanke(i).radius>0.015 && (norm(ich.pos-spielfeld.tanke(i)
            .pos)<(ich.radius+spielfeld.tanke(i).radius)) && system==1
90          f=-ich.ges;
91          if norm(ich.ges)<0.005 || norm(ich.pos-spielfeld.tanke(i).pos)<0.01
92              f=[0 0];
```



```
93          end
94          video.text='Stop!2';
95      end
96  end
```

Hierauf folgt nun das sehr umfassende Optimierungsverfahren. Dieses Funktion ermöglicht es unserem Spaceball **Minen**, die auf dem Weg zwischen uns und unserem Ziel liegen, zu umfahren.
Zu Anfang wird grundsätzlich abgefragt, ob unsere Geschwindigkeit größer als Null ist und in welchem Tanksystem wir uns gerade befinden, sowie dementsprechend die zugehörige Tankstelle zugewiesen.
Anschließend wird in einer for-Schleife abgefragt und berechnet, ob unser Spaceball auf dem Weg zur Tankstelle eine Mine schneiden würde. Wenn das der Fall ist, wird ein Ersatzpunkt bestimmt, der entweder rechts oder links von der Mine liegt. Dafür beziehen wir uns in unserer Berechnung auf den Geschwindigkeitsvektor unseres Spaceballs und weiterhin wird geprüft, über welchen Punkt der Weg um die Mine herum kürzer ist. Weiterhin wird geprüft, ob beide dieser Ersatzpunkte im Spielfeld liegen und in Zusammenhang mit unserer Größe überhaupt ansteuerbar und erreichbar sind. Wenn der eine Punkt diesen Kriterien nicht standhält, dann wird automatisch der andere Punkt ausgewählt, auch wenn dieser über den längeren Weg führt. Anschließend wird dieser Punkt in „Ziel" umbenannt und wir steuern direkt mit unserem Kraftvektor auf diesen zu. Dann folgt wie beim Tankstellenansteuerverfahren eine Berechnung zum Cosinus des Winkels mit dem unser Spaceball vom Kurs abweicht und entsprechende Gegensteuerungsmaßnahmen werden definiert.

```
97  if norm(ich.ges)>0
98      if system==1
99          Tanke=aktuelleTanke.pos;
100     else
101         Tanke=vglTanke.pos;
102     end
103     abbruch=0;
104     for k=0.001:0.03:100
105
106         neuepos=ich.pos+k*(Tanke-ich.pos);
107         if norm(neuepos-Tanke)<ich.radius || abbruch==1
108             break
109         end
110         for i=1:length(spielfeld.mine)
111             if norm(neuepos-spielfeld.mine(i).pos) < ich.radius+
                    spielfeld.mine(i).radius
112                 abbruch=1;
113                 a=ich.ges;
114                 b=[-a(2) a(1)];
115                 ir=(spielfeld.mine(i).radius+ich.radius+0.015)/norm(b);
116                 ad=b*ir;
117                 punkt1=(spielfeld.mine(i).pos+ad);
118                 punkt2=(spielfeld.mine(i).pos-ad);
119
120                 % beide Punkte liegen im Spielfeld
121                 if punkt1(1)>ich.radius+0.01 && punkt1(1)<1-ich.radius-0.01
                        && punkt1(2)>ich.radius+0.01 && punkt1(2)<1-ich.radius-0
```

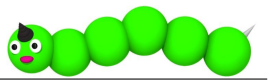


```
                        .01 ...
122                         && punkt2(1)>ich.radius+0.01 && punkt2(1)<1-
                                ich.radius-0.01 && punkt2(2)>ich.radius+0.01 &&
                                punkt2(2)<1-ich.radius-0.01
123                     if norm(ich.pos-punkt1)+norm(Tanke-punkt1) <= norm(
                            ich.pos-punkt2)+norm(Tanke-punkt2)
124                         ziel=punkt1;
125                     else
126                         ziel=punkt2;
127                     end
128                     % Punkt 1 liegt im Spielfeld
129                 elseif punkt1(1)>ich.radius+0.01 && punkt1(1)<1-ich.radius-0
                        .01 && punkt1(2)>ich.radius+0.01 && punkt1(2)<1-ich.radius
                        -0.01
130                     ziel=punkt1;
131                     % Punkt 2 liegt im Spielfeld
132                 else
133                     ziel=punkt2;
134                 end
135                 f=ziel-ich.pos;
136                 video.text='Umfahren';
137                 cosalpha3=dot(ich.ges, ziel-ich.pos)/(norm(ich.ges)*norm(ziel
                        -ich.pos));
138                 if cosalpha3<0.99 && norm(ich.ges)>0.01 && norm(ziel-
                        aktuelleTanke.pos)>0
139                     f=ziel-ich.pos-8*ich.ges;
140                     video.text='Korrektur 3';
141                 end
142             end
143         end
144
145     end
146 end
```

Im Anschluss an das Optimierungsverfahren folgen verschiedene Abbruchbedingungen.
Grundsätzlich bleibt unser Spaceball stehen, wenn alle Tankstellen auf dem Feld verbraucht sind. Um das herauszufinden wird eine beliebige Tankstelle wieder mit allen anderen verglichen und auf diese Weise herausgefunden, welche die größte Tankstelle derzeit ist.

```
147 abbruchTanke=spielfeld.tanke(randi([1 length(spielfeld.tanke)]));
148
149 for i=1:length(spielfeld.tanke)
150     tankei=spielfeld.tanke(i);
151     if tankei.radius>abbruchTanke.radius
152         abbruchTanke=spielfeld.tanke(i);
153     end
154 end
```

Wenn schließlich die größte noch vorhandene Tankstelle kleiner als 0.01 (Radius) ist und unser Spaceball größer ist als der des Gegners, dann wird der Schub abgestellt und wir fahren einfach weiter mit der Geschwindigkeit die wir haben.
Wenn unser Spaceball kleiner oder gleich groß ist als der des Gegners und sich keine



Minen auf dem direkten Weg zur aktuellen Tankstelle befinden, dann wird einmal kurz Schub gegeben um nicht zu viel Sprit zu verbrauchen und dieser dann sofort wieder ausgestellt, damit man sehr ökonomisch auf die Tankstelle zu gleiten kann.

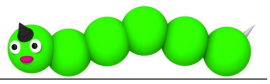
```
155 if abbruchTanke.radius<0.01 && ich.radius>gegner.radius
156     f=[0 0];
157     video.text='Ende';
158 elseif abbruchTanke.radius<0.01 && ich.radius<=gegner.radius && norm(ich.ges)
       <0.02 && mineImWeg==0
159     f=aktuelleTanke.pos-ich.pos;
160 elseif abbruchTanke.radius<0.01 && ich.radius<=gegner.radius && norm(ich.ges)
       >=0.02 && mineImWeg==0
161     f=[0 0];
162 end
```

Wenn es sich allerdings so verhält, dass wir über die Zeit des Spieles sehr gut tanken konnten und sogar so viel Sprit gespeichert haben, dass wir vor Ende bereits mehr davon besitzen, als der Gegner überhaupt mit den verbleibenden Tankstellen noch erreichen kann, dann bremsen wir erst einmal abrupt ab und bleiben stehen. Jetzt sind wir, also unser Spaceball, allerdings zu dick und zu behäbig um einen langsamen Angriff zu wagen, dann beginnen wir kontrolliert zu flackern, d.h. wir wollen unseren Spaceball die Geschwindigkeit Null erreichen lassen, weshalb er dann abwechselnd Schub in die eine und dann in die entgegengesetzte Richtung gibt. Das machen wir um Treibstoff zu verbrauchen und so wieder wendiger zu werden.

```
163 if ich.radius > gegner.radius ...
164         && ich.radius^2 >= gegner.radius^2 ...
165         + sum([spielfeld.tanke(1:length(spielfeld.tanke)).radius].^2)
166     f=-ich.ges;
167     if norm(ich.ges)<0.0005 && ich.radius > gegner.radius ...
168             && ich.radius^2 < 1.1*gegner.radius^2 ...
169             + sum([spielfeld.tanke(1:length(spielfeld.tanke)).radius].^2)
170         f=[0 0];
171         video.text='Treibstoff';
172     else
173         video.text='Treibstoff verbrauchen';
174     end
```

Jetzt folgt unser cooles System zum Anfahren von potentiellen letzten Tankstellen. Theoretisch hätten wir das Spiel jetzt bereits mit einem Punkt gewonnen, da wir größer als der Gegner sind. Um den Gegner nun noch vollständig mit Köpfchen zu schlagen, fahren wir auf eine noch recht große Tankstelle zu, die der Gegner auch ansteuern wird, und legen uns dort auf die Lauer. Dazu muss aber gegeben sein, dass der Weg zu ebendieser Tankstelle minenfrei ist, da sonst das Risiko zu hoch ist beim Umfahren viel Sprit zu verlieren. Auch hier verbrauchen wir „sinnlos" Treibstoff wenn wir am Zielort vor der Tankstelle angekommen sind um agiler zu werden.

```
175 if  minennummer(spielfeld,ich,ich.pos,aktuelleTanke.pos)==0 &&
       aktuelleTanke.radius>0.01
176         f=aktuelleTanke.pos-ich.pos;
177         if norm(ich.ges)^2>2*(4e-5/(ich.radius)^2)*(norm(ich.pos-
               aktuelleTanke.pos)-0.005-aktuelleTanke.radius-ich.radius)
178             f=-ich.ges;
```



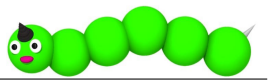
```
179             video.text='Treibstoff verbrauchen';
180             if norm(ich.ges)<0.0005 && ich.radius > gegner.radius ...
181                     && ich.radius^2 < 1.1*gegner.radius^2 ...
182                     + sum([spielfeld.tanke(1:length(spielfeld.tanke)).radius]
                            .^2)
183                 f=[0 0];
184                 video.text='Treibstoff';
185             end
186         end
187     end
```

Wenn der Gegner ebenso wie wir nicht mehr beschleunigt, also aufgehört hat zu tanken und uns nicht angreift, sondern nur noch durch die Gegend gleitet, dann untersuchen wir weiterhin ob der Weg zum Gegner nicht durch eine Mine blockiert ist und wir eine moderate Größe zum Angreifen besitzen. Sollten alle diese Voraussetzungen gegeben sein beschleunigen wir auf eine Geschwindigkeit von 0.05, stellen dann den Schub wieder ab und gleiten langsam auf den Gegner zu.
Jetzt stellt sich die Frage, warum gibt es diesen Teil des Codes doppelt? Nun, wir wissen es selbst nicht so genau, vermutlich hatten wir versucht dieses Verfahren noch zu optimieren, was wohl nicht funktioniert hat und es ist einfach doppelt stehen geblieben. Mit einem kleinen Augenzwinkern sei also gesagt: Weil das Verfahren so schön ist, ist es doppelt enthalten.

```
188     if  minennummer(spielfeld,ich,ich.pos,gegner.pos)==0 && norm(gegner.bes)
            <=0.005 ...
189             && ich.radius > gegner.radius ...
190             && ich.radius^2 > 1.1*gegner.radius^2 ...
191             + sum([spielfeld.tanke(1:length(spielfeld.tanke)).radius].^2)
192         if norm(ich.ges)<0.05
193             f=gegner.pos-ich.pos+gegner.ges;
194         elseif norm(ich.ges)>=0.05
195             f=[0 0];
196         end
197         video.text='langsamer Angriff';
198     elseif minennummer(spielfeld,ich,ich.pos,gegner.pos)==0 && norm(
            gegner.ges)<=0.005 ...
199             && ich.radius > gegner.radius ...
200             && ich.radius^2 > 1.1*gegner.radius^2 ...
201             + sum([spielfeld.tanke(1:length(spielfeld.tanke)).radius].^2)
202         if norm(ich.ges)<0.05
203             f=gegner.pos-ich.pos+gegner.ges;
204         elseif norm(ich.ges)>=0.05
205             f=[0 0];
206         end
207         video.text='langsamer Angriff';
208     end
```

Sollten wir allerdings schon sehr nah an den Gegner herangerückt sein und er nicht mehr weit entfernt von uns sein, dann starten wir einen schnellen Angriff um ihn in die Enge zu treiben.

```
209     if norm(ich.pos-gegner.pos)-ich.radius-gegner.radius<0.075
210         f=gegner.pos-ich.pos+gegner.ges;
211         das=ka(ich,gegner);
212         if isempty(das.kreis1pos1)==0
```



```
213             f=das.kreis1pos1-ich.pos;
214             video.text='Treibangriff';
215         end
216     end
217 end
```

Als nächster Teil im Code folgt nun das **Anhaltesystem** für die Tankstellen. Mit der Ansteuerung ebendieser und dem zugehörigen Korrekturverfahren hatten wir uns ja bereits beschäftigt, hier folgt nun die Schubdefinition wenn der Spaceball sich jetzt schon sehr nahe an einer Tankstelle befindet. Hier sind grundsätzlich wieder System 1 und 2 zu unterscheiden. Zuerst wird sich mit dem System 1 beschäftigt, also beim Zuhalten auf eine einzelne Tankstelle. Dann bremsen wir stark ab und stellen sobald eine sehr geringe Geschwindigkeit erreicht ist den Schub sogar ganz ab.

```
218 if norm(ich.ges)^2>2*(4e-5/(ich.radius)^2)*(norm(ich.pos-aktuelleTanke.pos)-0
      .025) && tanken==1 && system==1
219     f=-ich.ges;
220     if norm(ich.ges)<0.005
221         f=[0 0];
222     end
223     video.text='Anhalten!';
224 end
```

Wenn System 2 angeschaltet ist und wir uns dementsprechend auf dem Weg zum Mittelpunkt zwischen 2 Tankstellen befinden, dann bremsen und verhalten wir uns äquivalent.

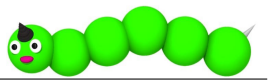
```
225 if system==2 && norm(ich.ges)^2>2*(4e-5/(ich.radius)^2)*(norm(ich.pos-((
      aktuelleTanke.pos+vglTanke.pos)/2))-0.025)
226     f=-ich.ges;
227     if norm(ich.ges)<0.005
228         f=[0 0];
229     end
230     video.text='Mittelpunkt-Anhalten';
231 end
```

Das Kernelement zum 2-Punkte-Sieg ist der **„Angriff"**. Unser Angriff im speziellen basiert auf der Kollisionsanalyse zweier Kreise, die im weiteren Verlauf der kraft-Datei noch näher erläutert wird. Wie der Name es bereits durchblicken lässt wird hier ausgerechnet wann und wo die beiden Spaceballs einander schneiden. Deshalb steht am Anfang unseres Angriffs auch die Kollisionsanalyse:

```
232 s=ka(ich,gegner);
233 s.kreis1pos1;
234 ich.pos;
```

Darauf folgen die Bedingungen wann angegriffen werden soll: Natürlich nur wenn wir größer als der Gegner sind und allgemein einigermaßen groß, wenn wir uns nicht nahe an einer Tankstelle befinden oder wenn wir uns sehr nahe am Gegner befinden und nur ein wenig größer als er sind.

```
235 if ich.radius>1.15*gegner.radius...
236         && ich.radius>0.035...
```



```
237         && norm(ich.pos-aktuelleTanke.pos)>0.1...
238         ||ich.radius>1.05*gegner.radius...
239         && ich.radius>0.035...
240         && norm(ich.pos-gegner.pos)<0.1...
241         && norm(ich.pos-aktuelleTanke.pos)>0.1
```

Darauf folgen die verschiedenen Angriffsvarianten. Am wichtigsten ist hierbei wie weit wir vom Gegner entfernt sind, aber auch wie schnell wir sind oder wie schnell der Gegner sich fortbewegt und vor allem in welche Richtung er sich bewegt. Je nachdem wie diese Kriterien ausfallen, gibt es verschiedene Modi. Alle treten nur dann in Kraft, wenn auf dem Weg keine Mine liegt.

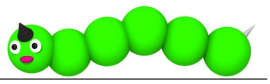
```
242     if mineImWeg==0
243 
244         if isempty(s.la1)==0 && 0<s.la1<3 && 0<s.la2<3 && norm(ich.pos-
                gegner.pos)<0.4
245 
246             f=s.kreis1pos1-ich.pos;
247             video.text='Angriff';
248         elseif isempty(s.la1)==1 && norm(ich.pos-gegner.pos)<0.05
249 
250             f=gegner.pos-ich.pos;
251             video.text='Angriff';
252         elseif isempty(s.la1)==0 && norm(ich.pos-gegner.pos)*1.1<norm(ich.pos
                -s.kreis1pos1) && norm(gegner.ges)<0.05
253 
254             f=gegner.pos-ich.pos;
255             video.text='Angriff';
256         elseif isempty(s.la1)==0 && 0>s.la1 && 0>s.la2 && norm(ich.pos-
                gegner.pos)<0.05
257 
258             f=gegner.pos-ich.pos;
259             video.text='Angriff';
260         end
261     end
262 end
```

Nach dem Angriff folgt das Pendant: die **Verteidigung**. Diese haben wir sehr schlicht und eher klein gehalten, denn wenn man - so unsere Philosophie - sehr gut und effektiv tankt, nicht in Minen hineinfährt und sich einigermaßen ökonomisch fortbewegt, dann gibt es das Problem, dass man vom Gegner attackiert wird, schlichtweg einfach nicht.
Unserer Verteidigung liegt auch wieder eine Kollisionsanalyse zugrunde, die als erstes durchgeführt wird:

```
263 v=ka(ich,gegner);
```

Wenn der Gegner dann noch größer ist als unser Spaceball und eine Kollision kurz bevor steht, dann entfernen wir uns in eine andere Richtung.

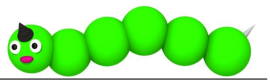
```
264 if gegner.radius>ich.radius*1.05 && isempty(v.la1)==0 && 0<v.la1<3 && 0<v.la2
        <3
265 
```



```
266     f=v.kreis1pos1+gegner.ges-ich.pos;
267     video.text='Verteidigung';
268 end
```

In den folgenden Zeilen taucht nun eines unserer wichtigsten Schutzmechanismen auf: das **Wandpräventationssystem**. Hierbei wird für jede Seite im Spielfeld und jede Ecke analysiert, ob unser Spaceball in diese hineinrasen könnte und ein entsprechender flag gesetzt. Dazu wird unser Geschwindigkeitsvektor in Komponenten zerlegt und mit den relevanten Komponenten der entsprechende Bremsweg zur Wand berechnet. Wenn der Abstand zur Wand gerade noch größer als der Bremsweg ist wird in die Gegenrichtung der entsprechenden Geschwindigkeitskomponente gesteuert. In den Ecken müssen logischerweise mehrere Komponenten betrachtet werden.

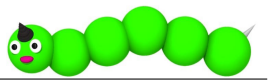
```
270 if (ich.ges(1)>0 && ich.ges(1)^2>=ich_bes_max*(ich_abstand_rechts)) || 1-
    ich.pos(1)<ich.radius+0.01
271     % rechts
272     flag_bande_rechts=1;
273     video.text='rechte Bande';
274 end
275
276 if (ich.ges(1)<0 && ich.ges(1)^2>=ich_bes_max*(ich_abstand_links)) || ich.pos
    (1)<ich.radius+0.01
277     % links
278     flag_bande_links=1;
279     video.text='linke Bande';
280 end
281
282 if (ich.ges(2)<0 && ich.ges(2)^2>=ich_bes_max*(ich_abstand_oben)) || ich.pos
    (2)<ich.radius+0.01
283     % oben
284     flag_bande_oben=1;
285     video.text='obere Bande';
286 end
287
288 if (ich.ges(2)>0 && ich.ges(2)^2>=ich_bes_max*(ich_abstand_unten)) || 1-
    ich.pos(2)<ich.radius+0.01
289     % unten
290     flag_bande_unten=1;
291     video.text='untere Bande';
292 end
293
294 if ((ich.ges(1)>0 && ich.ges(1)^2>=ich_bes_halb*(ich_abstand_rechts))&&(
    ich.ges(2)<0 && ich.ges(2)^2>=ich_bes_halb*(ich_abstand_oben)))
295     % oben rechts
296     flag_bande_obenrechts=1;
297 end
298
299 if ((ich.ges(1)>0 && ich.ges(1)^2>=ich_bes_halb*(ich_abstand_rechts))&&(
    ich.ges(2)>0 && ich.ges(2)^2>=ich_bes_halb*(ich_abstand_unten)))
300     % unten rechts
301     flag_bande_untenrechts=1;
302 end
```



```
303 if ((ich.ges(1)<0 && ich.ges(1)^2>=ich_bes_halb*(ich_abstand_links))&&(
        ich.ges(2)>0 && ich.ges(2)^2>=ich_bes_halb*(ich_abstand_unten)))
304     % unten links
305     flag_bande_untenlinks=1;
306 end
307 if ((ich.ges(1)<0 && ich.ges(1)^2>=ich_bes_halb*(ich_abstand_links))&&(
        ich.ges(2)<0 && ich.ges(2)^2>=ich_bes_halb*(ich_abstand_oben)))
308     % oben links
309     flag_bande_obenlinks=1;
310 end
```

Daran schließt sich unser **Minensystem** an. Hier ist vor allem von Interesse, dass zwei nicht durchfahrbare Minen durch eine dritte imaginäre Mine ersetzt werden, die mit ihrer Fläche die beiden anderen Minen umschließt und dann großräumig umfahren wird.
In diesem Zusammenhang schauen wir uns auch an, ob wir auf eine normale oder imaginäre/zusätzliche Mine zufahren. Dann wird natürlich gebremst, damit wir nicht in die Minen hineinfahren. Weiterhin wird festgelegt, dass unser Spaceball, wenn er nur noch einen sehr kleinen Abstand zu der Mine hat, Schub von der Mine weg gibt. Damit wird vor allem verhindert, dass er beim Tanken in eine Mine hineinwächst.

```
311 for h=1:length(spielfeld.mine)
312     for g=1:length(spielfeld.mine)
313         if g>h && norm(spielfeld.mine(h).pos-spielfeld.mine(g).pos)<
                spielfeld.mine(h).radius+spielfeld.mine(g).radius+2*ich.radius+0
                .03
314             zusatzmine.pos=(spielfeld.mine(h).pos+spielfeld.mine(g).pos)/2;
315             zusatzmine.radius=((norm(spielfeld.mine(h).pos-spielfeld.mine(g)
                    .pos))+spielfeld.mine(h).radius+spielfeld.mine(g).radius)/2;
316
317             winkelzupunkt=atan((zusatzmine.radius+ich.radius+0.01)/norm(
                    ich.pos-zusatzmine.pos));
318             winkeligeszumine=acos(dot(ich.ges, zusatzmine.pos-ich.pos)/(norm(
                    ich.ges)*norm(zusatzmine.pos-ich.pos)));
319
320             if norm(ich.ges)^2>=2*(4e-5/(ich.radius)^2)*(norm(zusatzmine.pos-
                    ich.pos)-ich.radius-zusatzmine.radius-0.01) ...
321                     && winkeligeszumine-winkelzupunkt<0 && norm(
                            zusatzmine.pos-ich.pos)>norm(aktuelleTanke.pos-ich.pos
                            )...
322                     && norm(ich.pos-aktuelleTanke.pos)>norm(ich.pos-
                            zusatzmine.pos)
323                 f=-ich.ges;
324             elseif(norm(zusatzmine.pos-ich.pos)-zusatzmine.radius-ich.radius)
                    <0.009 && norm(zusatzmine.pos-ich.pos)>norm(aktuelleTanke.pos-
                    ich.pos)...
325                     && norm(ich.pos-aktuelleTanke.pos)>norm(ich.pos-
                            zusatzmine.pos)
326                 f=ich.pos-zusatzmine.pos;
327             end
328         end
329     end
330 end
331
```



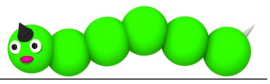
```
332 for i=1:length(spielfeld.mine)
333     minei=spielfeld.mine(i);
334     winkelzupunkt=atan((minei.radius+ich.radius+0.01)/norm(ich.pos-minei.pos)
            );
335     winkeligeszumine=acos(dot(ich.ges, minei.pos-ich.pos)/(norm(ich.ges)*norm
            (minei.pos-ich.pos)));
336
337     if norm(ich.ges)^2>=2*(4e-5/(ich.radius)^2)*(norm(minei.pos-ich.pos)-
            ich.radius-minei.radius-0.01) ...
338             && winkeligeszumine-winkelzupunkt<0
339         f=-ich.ges;
340     elseif norm(minei.pos-ich.pos)-minei.radius-ich.radius<0.009
341         f=ich.pos-minei.pos;
342     end
343 end
```

Weiterhin muss natürlich noch die flag-Auswertung vorgenommen werden, wo je nachdem auf jede Wand gerade zugesteuert wird entsprechender Gegenschub erzeugt wird.

```
344 if(flag_bande_rechts) f=[-1 0]; end
345 if(flag_bande_links) f=[1 0]; end
346 if(flag_bande_oben) f=[0 1]; end
347 if(flag_bande_unten) f=[0 -1]; end
348 if(flag_bande_obenrechts) f=[-1 1]; end
349 if(flag_bande_untenrechts) f=[-1 -1]; end
350 if(flag_bande_untenlinks) f=[1 -1]; end
351 if(flag_bande_obenlinks) f=[1 1]; end
352
353 end
```

Das Weiteren folgen noch **wichtige functions**, auf die im Laufe unserer Berechnungen immer wieder zugegriffen wird.
Zum einen ist das die function **„aktuelleTankeBestimmen"**, welche - man ahnt es vielleicht bereits - dazu dient eine aktuelle Tankstelle auszuwählen, auf die dann zugefahren wird.
Durch Zufall wird hier eine bestimmte Tankstelle ausgewählt und dann mit den übrigen bezüglich verschiedener Kriterien verglichen, welche eine unterschiedliche Wertigkeit besitzen, die mit verschieden großen Faktoren in die Kalkulation eingehen. Das sind zum einen Nähe und Größe der Tankstelle, zum anderen aber auch wie weit der Gegner von der Tankstelle entfernt ist, oder ob er sie bereits berührt.

```
354 function [ aktuelleTanke ] = aktuelleTankeBestimmen (spielfeld, ich, gegner)
355 aktuelleTanke=spielfeld.tanke(randi([1 length(spielfeld.tanke)]));
356
357 for i=1:length(spielfeld.tanke)
358     tankei=spielfeld.tanke(i);
359     if tankei.radius>0.005 && norm(ich.pos-tankei.pos)/2-norm(gegner.pos-
            tankei.pos)/5-10*tankei.radius ...
360             <= norm(ich.pos-aktuelleTanke.pos)/2-norm(gegner.pos-
                aktuelleTanke.pos)/5-10*aktuelleTanke.radius
361         if norm(tankei.pos-gegner.pos)>tankei.radius+gegner.radius
362             aktuelleTanke=spielfeld.tanke(i);
```



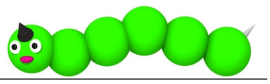
```
363         elseif norm(tankei.pos-gegner.pos)<tankei.radius+gegner.radius
364             if gegner.radius*1.15<ich.radius
365                 aktuelleTanke=spielfeld.tanke(i);
366             end
367         end
368     end
369 end
370
371 end
```

Weiterhin existiert eine function **„minennummer"**, welche ermittelt, ob sich zwischen Startpunkt und Endpunkt, also beispielsweise alte Tankstelle und neue Tankstelle, eine Mine befindet. Sollte das der Fall sein, dann wird die Nummer ebendieser Mine ausgegeben. Sollte keine Mine im Weg sein, wird die Minennummer 0 ausgegeben. Auf diese Nummer wird dann unter anderem im Minenumfahrsystem zurückgegriffen.

```
372 function [ minennummer ] = minennummer(spielfeld,ich,SP,EndP)
373 minennummer=0;
374 for k=0.001:0.05:100
375     neuepos=SP+k*(EndP-SP);
376     if norm(neuepos-EndP)<0.05
377         break
378     end
379     for i=1:length(spielfeld.mine)
380         if norm(neuepos-spielfeld.mine(i).pos) < ich.radius+spielfeld.mine(i)
                .radius && minennummer==0
381             minennummer=i;
382         end
383     end
384 end
385
386 end
```

Last but (not) least findet sich im kraft-file schlussendlich noch die **Kollisionsanalyse** zweier Kreise, also unserer Spaceballs untereinander oder unseres Spaceballs mit einer Mine. Hierbei wird nicht nur der Ort berechnet, an welchem sich die beiden Spaceballs berühren werden, sondern auch wann. Das ermöglicht ein großes Maß an strategischer Planung.

```
387 function s=ka(kreis1, kreis2)
388
389 oa=kreis1.pos;
390 a=kreis1.ges;
391
392 oc=kreis2.pos;
393 b=kreis2.ges;
394
395 e=oc-oa;
396 f=b-a;
397
398 R12=kreis1.radius+kreis2.radius;
399
400 alpha=dot(f,f);
```



```
401 beta=dot(e,f);
402 gamma=dot(e,e)-R12^2;
403
404 delta=beta^2-alpha*gamma;
405
406 if delta>0
407     la1=(-beta+sqrt(delta))/alpha;
408     la2=(-beta-sqrt(delta))/alpha;
409
410     [la]=sort([la1 la2]);
411
412     la1=la(1);
413     la2=la(2);
414
415     kreis1pos1=oa+la1*a;
416     kreis1pos2=oa+la2*a;
417
418     kreis2pos1=oc+la1*b;
419     kreis2pos2=oc+la2*b;
420
421 else
422     la1=[ ];
423     la2=[ ];
424     kreis1pos1=[ ];
425     kreis2pos1=[ ];
426     kreis1pos2=[ ];
427     kreis2pos2=[ ];
428 end
429
430 field1 = 'la1';  value1 = la1;
431 field2 = 'la2';  value2 = la2;
432 field3 = 'kreis1pos1';  value3 = kreis1pos1;
433 field4 = 'kreis1pos2';  value4 = kreis1pos2;
434 field5 = 'kreis2pos1';  value5 = kreis2pos1;
435 field6 = 'kreis2pos2';  value6 = kreis2pos2;
436
437 s = struct(field1,value1,field2,value2,field3,value3,field4,value4,field5,
       value5,field6,value6);
438
439 end
```

5 Dokumentation der Animation

5.1 1. Projektwoche

Nachdem wir uns als Gruppe darüber abgestimmt hatten, dass wir als Spaceball eine Raupe haben möchten, konnte auch die Animation langsam voran schreiten. Der erste Arbeitsschritt war hierbei natürlich das Bearbeitungsprogramm „Maya Autodesk" (Version 2014) herunterzuladen. Hierbei handelt es sich um eine Software für 3-D-Modellierung und Animation. Es zeichnet sich durch eine hohe visuelle Qualität aus und bietet einen umfassenden Editor zum erstellen von eigenen Objekten und Szenen. Der nächste Arbeitsschritt bestand dann darin sich in Maya einzuarbeiten. Dies geschah mithilfe verschiedener Videotutorials von Youtube und durch schlichtes Ausprobieren.
Dann konnte ein erster Entwurf der Raupe entstehen. Als Grundlage für den Körper dienen hierbei fünf ineinandergeschobene grüne Kugeln und das Horn wird mithilfe einer Pyramide animiert.

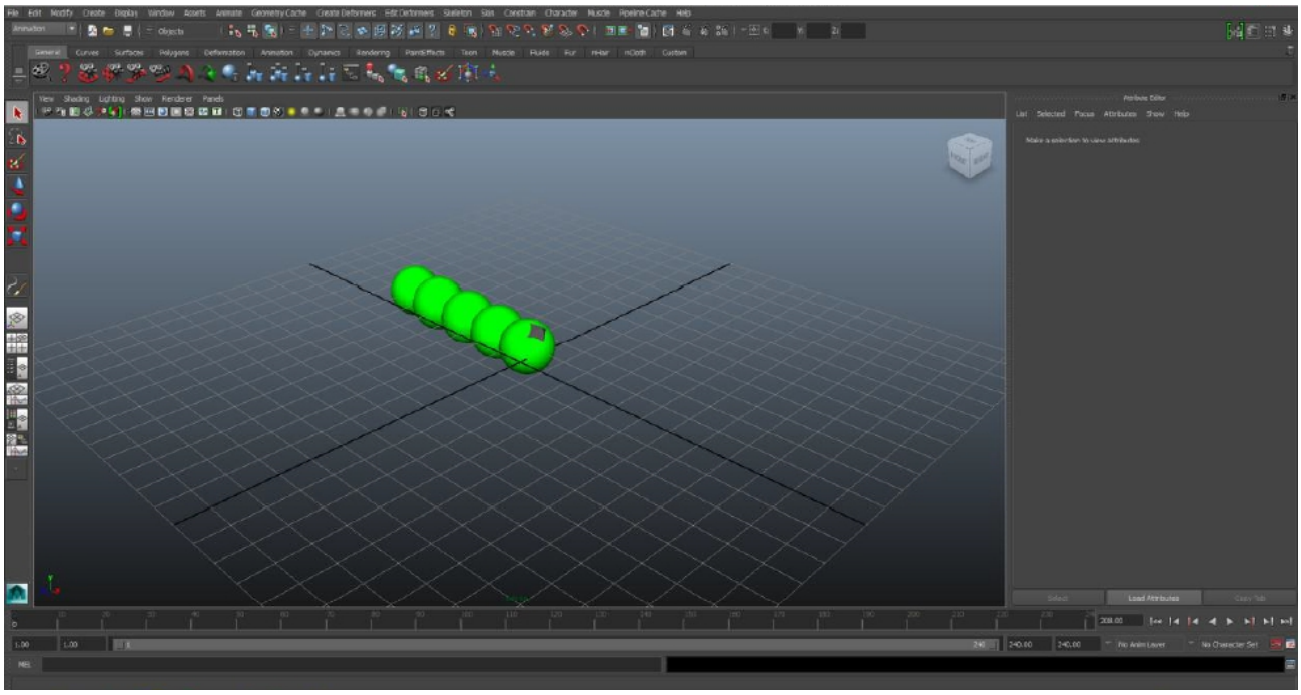


Abbildung 5: Körper von KillerRaupi

Des Weiteren wurde die Farbgebung der Raupe optimiert und harmonischer gestaltet. Außerdem haben wir angefangen KillerRaupi in Bewegung zu setzen. Das bestand erst einmal darin dass unsere Raupi einem vorgegebenen Pfad (einer geraden Linie) erfolgreich folgt.
Im Weiteren Verlauf bekam die Raupe dann auch noch Augen und einen Mund, wie die Abbildung auf der folgenden Seite zeigt.

Für die nächste Woche wird angestrebt, dass KillerRaupi ein noch schöneres Aussehen erhält (Augen, Mund, Nase), sowie ein vielleicht erstes vorläufiges Video erstellt werden kann.

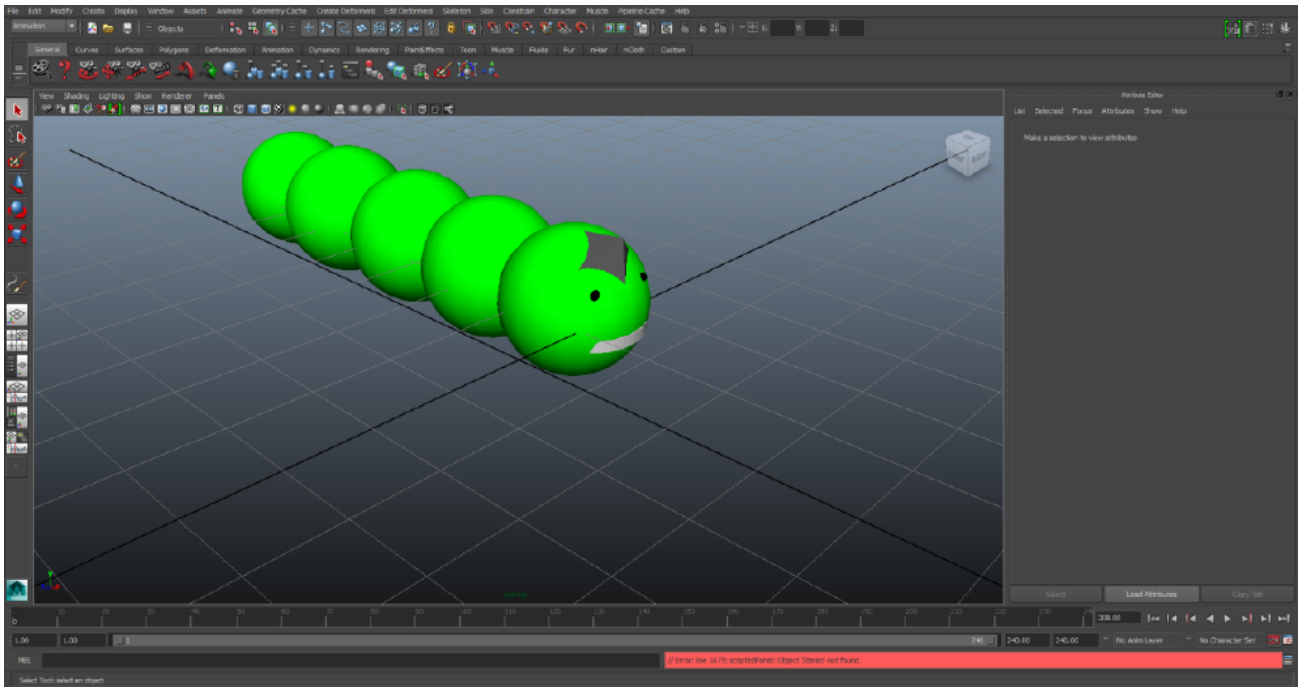


Abbildung 6: Raupi mit Augen und Mund

5.2 2. Projektwoche

In dieser Woche sollte das Video erstellt werden, welches eingeblendet wird, wenn der Spaceball eine Mine touchiert. Dafür schwebte uns vor, dass KillerRaupi auf eine Mine zufährt, diese berührt und man dann eine große Explosion sieht.
Diese Explosion erstellten wir hierbei mithilfe eines Video-Tutorials, das Resultat ist in der folgenden Abbildung zu sehen:

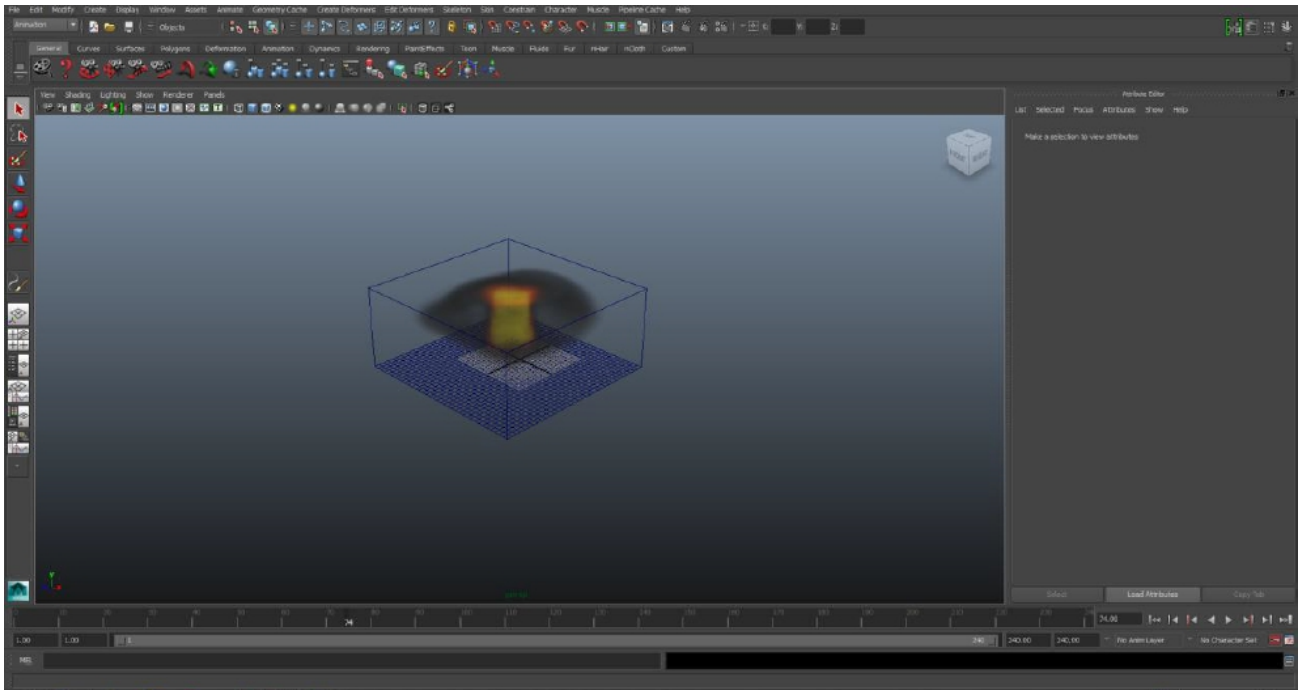


Abbildung 7: Erste Fassung einer Explosion

Des Weiteren wurde das Aussehen von KillerRaupi verbessert: Sie erhielt einen Kegel als Horn, sowie einen Stachel am Körperende, zwei größere Kugeln als Augen und eine stark deformierte rote Kugel als Nase. Dies geschah, falls dieses Grundaussehen irgendwoher bekannt sein sollte, in Anlehnung an ein Pokemon.

In der folgenden Abbildung ist auch bereits eine Mine (ein kleines graues Oval vor Raupi) zu sehen, für das Video, in welchem KillerRaupi auf eine Mine trifft.
Diese Szene, wie Raupi auf die Mine „zufährt" wurde für das Video zuerst aufgenommen. Daran schließt sich die Aufnahme der Szene wenn KillerRaupi explodiert an. Anschließend wurde das gesamte Video durch Verknüpfen diese beiden Szenen erstellt. Zu diesem Zweck brauchten wir noch ein anderes Programm, weshalb wir Windows Movie Maker heruntergeladen haben. So konnte unser erstes KillerRaupi-Video[2] mit einer stolzen Dauer von 4 Sekunden entstehen.

Des Weiteren wurden noch weitere 3-D-Animationen erstellt, die auf der Website für Farbe und Anschaulichkeit sorgen sollten.

[2]Hinweis: Alle Videos sind zum Anschauen auf unserer Website unter http://killerraupi.wix.com/killerraupi#!animation/caxr zu finden

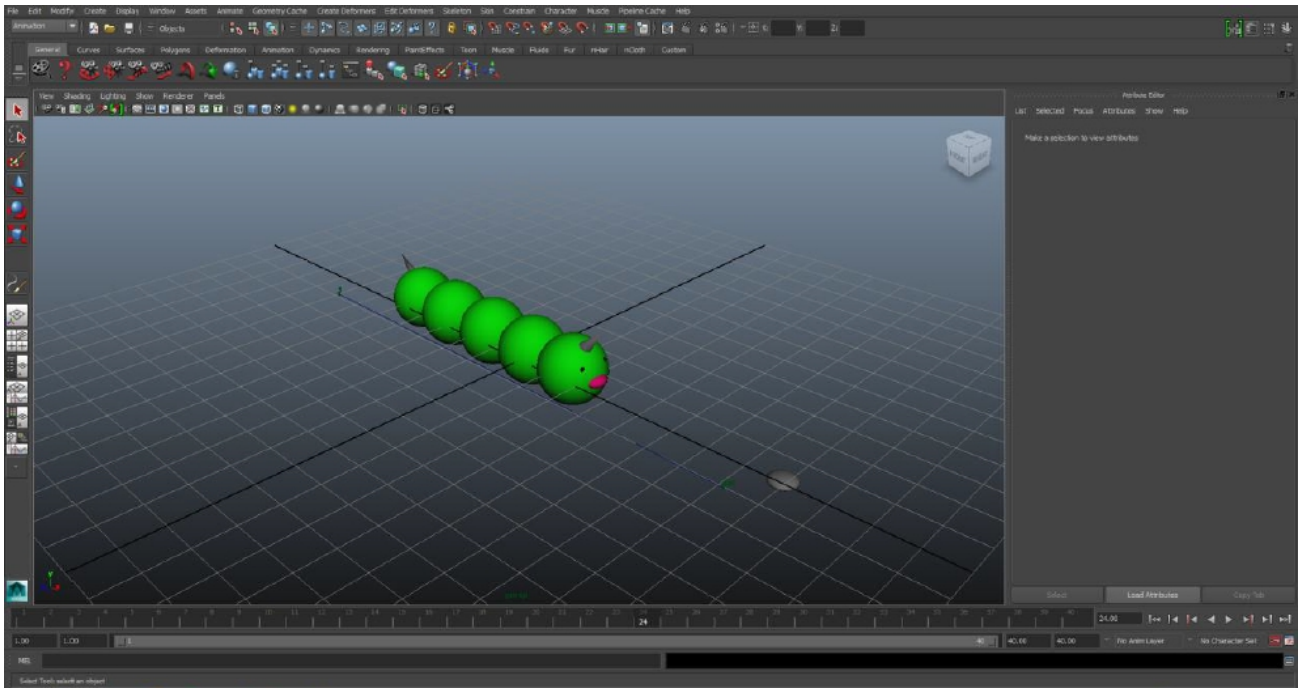


Abbildung 8: Hübsche KillerRaupi nahe Mine

5.3 3. Projektwoche

Die Hauptarbeit dieser Woche für die Animation bestand darin eine anschauliche und attraktive Minipräsentation zu entwerfen, um vor dem betreuenden Professor bestehen zu können. Das gelang soweit auch sehr gut und wir konnten uns dem Erstellen der Videos widmen.
Als nächstes wird das Video entstehen, wie KillerRaupi auf eine Bande zufährt und mit dieser kollidiert. Uns schwebt derzeit vor, dass die Raupe gegen die Wand fährt und sich daran „zusammenschiebt" und am Ende nur ein runder grüner Klecks von Raupi übrig bleibt. Allerdings wurde diese Woche erst einmal nur der Teil des Videos animiert, in welchem die Raupe auf die Wand zufährt. Das Resultat ist in der nächsten Abbildung dargestellt.

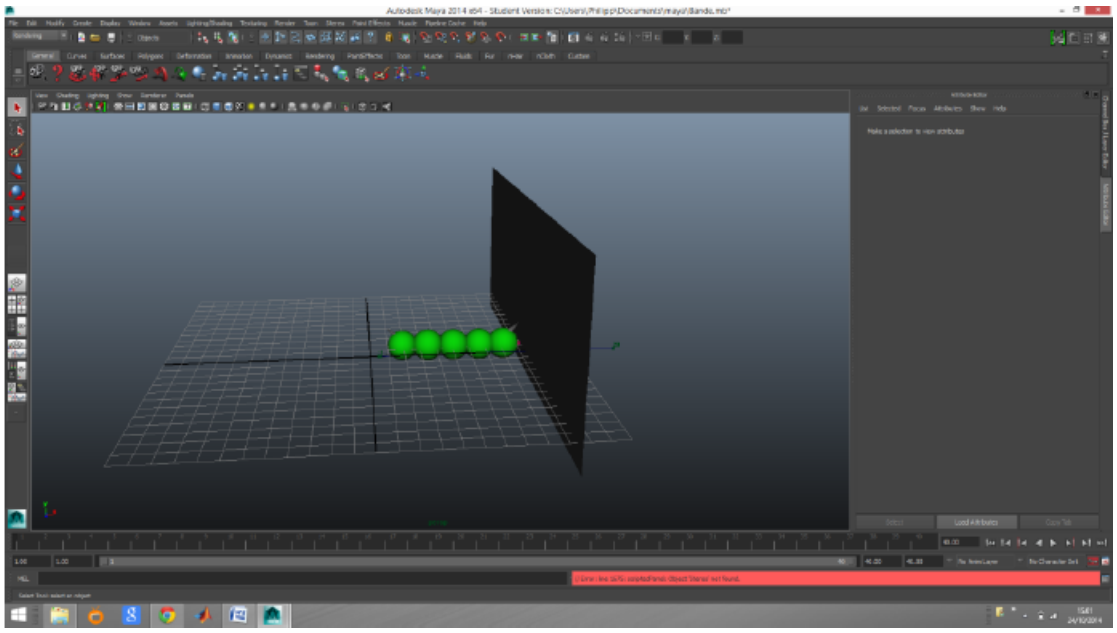


Abbildung 9: Erster Teil des Videos „Wand"

Des Weiteren wurde das Aussehen von KillerRaupi noch entscheidend verbessert. Anstatt des einfachen Kegels als Horn erhielt KillerRaupi nun ein gedrehtes Horn, ähnlich dem eines Einhorns. Außerdem war uns KillerRaupi noch nicht niedlich genug, weshalb sie große Kulleraugen erhalten hat, die ihr ein liebenswertes Aussehen sichern.
Dieses Bild fand diese Woche damit auch den Eingang in die Kopfzeile der Dokumentation.

Abbildung 10: süße KillerRaupi

5.4 4. Projektwoche

In der vergangenen Woche hatte das Animationsteam mit der Animation eines Videos begonnen, in dem KillerRaupi an der Wand zerdrückt wird, wenn sie gegen eine Bande fährt. Diese Szene wurde nun diese Woche noch hinzugefügt:

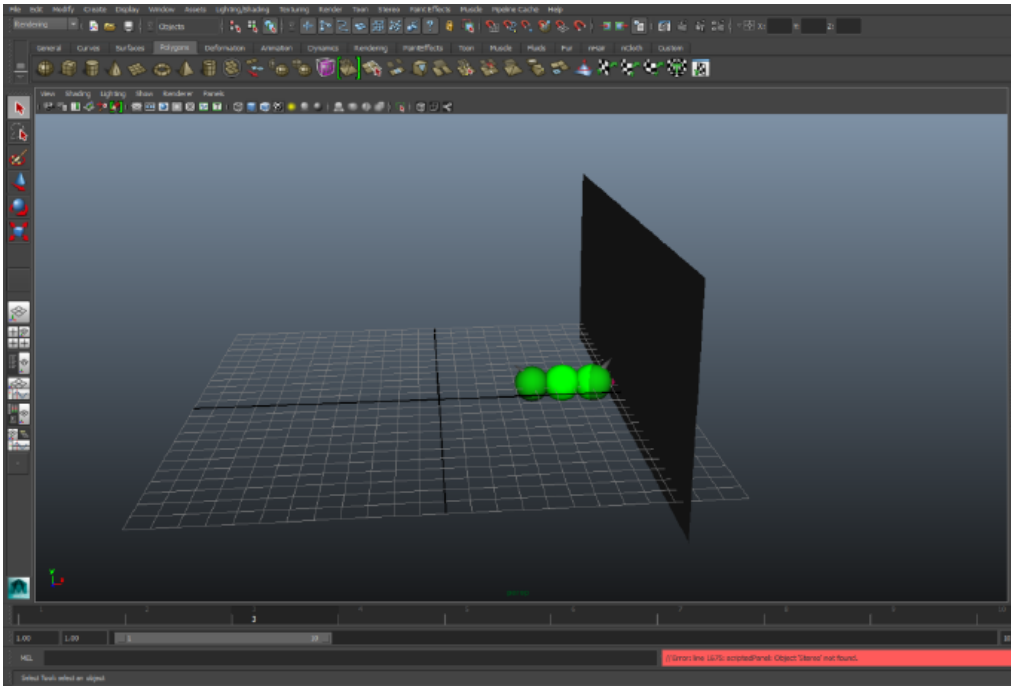


Abbildung 11: Szene: KillerRaupi wird an der Wand zerdrückt

Anschließend wurden die Einzelszenen mit Windows Movie Maker zu einem kompletten Clip zusammen gesetzt.

Des Weiteren wurde diese Woche das bestehende Video „Explosion an der Mine" verbessert, indem die Kameraperspektive beim Zufahren auf die Mine und bei der Explosion selbst verändert wurde und damit Details besser zu erkennen sind.

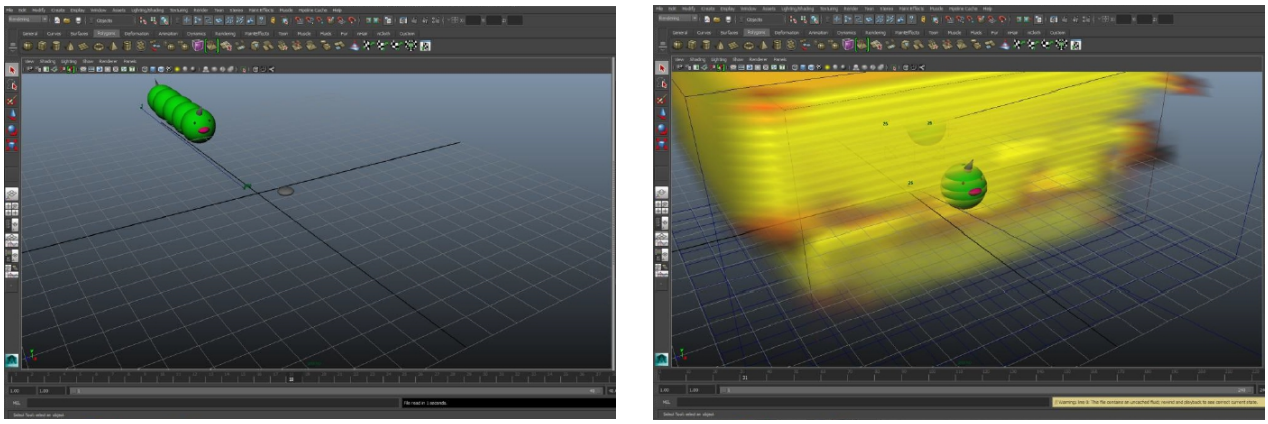


Abbildung 12: Verbesserter Clip Explosion

5.5 5. Projektwoche

In dieser Woche hat sich eine neue Idee für ein Video durchgesetzt: Wenn KillerRaupi verliert (was hoffentlich nie passiert), dann geht sie nach Hause in ihr Haus, knallt die Tür zu und zieht sich zurück. Um das realisieren zu können, braucht KillerRaupi allerdings erst einmal ein Haus. Dieses Miniprojekt stand diese Woche im Zentrum der Animationsarbeit.

Nachdem man sich mit den grundlegenden Techniken des „Hausbaus" in Maya beschäftigt hatte, konnte das Haus einen Grundriss mit senkrechten Wänden und einen Türrahmen erhalten. Dieser besteht aus einem gebogenen Zylinder und wurde quasi mit der Vorderseite des Hauses verschmolzen. Hierbei war es notwendig auch einzelne „faces", also Flächen, auf die bereits bestehenden Oberflächen zu ziehen, damit diese dann später einzeln gefärbt werden können.
Nach diesem Prinzip erhielt die Tür noch zwei Planken als nette Details und natürlich einen Türgriff.
Außerdem wurden „supporting edges" eingefügt, ein Tool welches die Ecken und Kanten eines Körper beliebig stark runden kann. Somit wurde das Aussehen von Raupis Haus insgesamt etwas weicher und niedlicher.

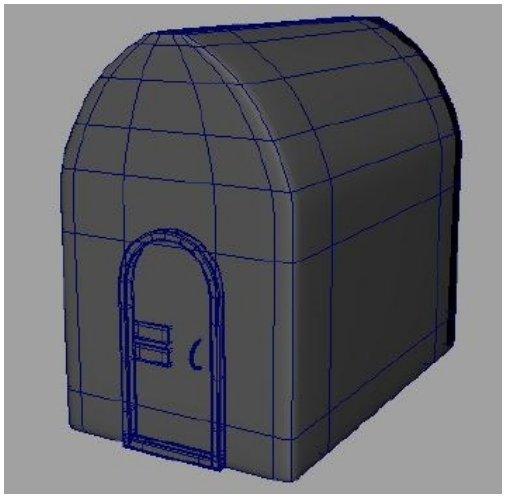


Abbildung 13: Rohbau von Raupis Haus

Als nächstes erhielt Raupis Haus ein Dachfenster inkl. einer Fensterbank und es wurden noch weitere „backplanes" zum Färben des Hauses angebracht, da man sonst nicht spezielle Flächen allein färben könnte. Anschließend wurde das Dach und das rechte Fenster dupliziert. Des Weiteren bekam das Haus einige Verzierungen an seinen Ecken, die für einen comichaften Effekt sorgen sollen und bei der Animation des Vidoes für einen „Boost-Effekt" beim Türknallen sorgen sollen.
Für alle 4 Hausseiten wurden nun die Details und Fenster dupliziert, was allerdings dazu führte, dass unser Haus ausgesprochen künstlich und wenig niedlich wirkte. Deshalb musste jede Hausseite noch mit kleinen Details individualisiert werden.

Als nächster Arbeitsschritt sollte das Dach gedeckt werden. Wie in der Realität braucht man dazu Dachziegel. In der nebenstehenden Abbildung ist solch eine Prototyp-Dachziegel zu sehen, ebenso wie ein Zylinder, der in die Aussparungen der Dachziegel eingesetzt wurde, damit sie ein möglichst realistisches Aussehen besitzen. Dieser eine Dachziegel wurde dann über die gesamte Reihe dupliziert, dann wurden die Reihen abwechselnd gespiegelt und einzelne Ziegel noch gedreht und gewendet, damit es auch hier nicht wie konstruiert aussieht.

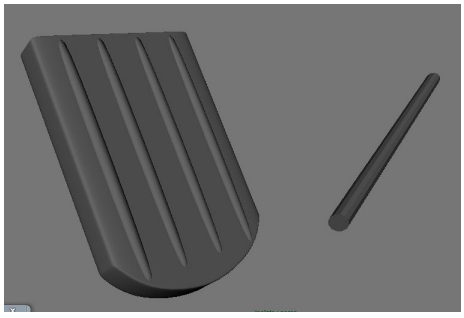


Abbildung 14: Dachziegel mit Welle

Danach wurde die eine Seite des Daches dupliziert und auf die andere Seite übertragen. Schließlich brauchte das Haus nur noch einen Schornstein (bestehend aus einem Quader und einem gebogenen Zylinder) und noch ein wenig Tageslicht, viel Farbe und fertig war KillerRaupis Zuhause.

Abbildung 15: Raupis Haus

5.6 6. Projektwoche

Diese Woche ist wieder ein neues Video entstanden, allerdings leider wieder für einen negativen Ausgang des Spieles für KillerRaupi. Sollte einmal der unwahrscheinliche Fall eintreten, dass KillerRaupi keine Tankstelle mehr findet und verschwindend klein wird, dann wird das im Video gezeigt, indem man die Raupe auf die ersehnte Tankstelle, also Trinkflasche, zufahren sieht und sie es aber nicht mehr schafft und schließlich einfach verschwindet. Aus diesen beiden Sequenzen besteht auch das Video, welches dann wie bereits zuvor im Windows Movie Maker zusammengesetzt wird.

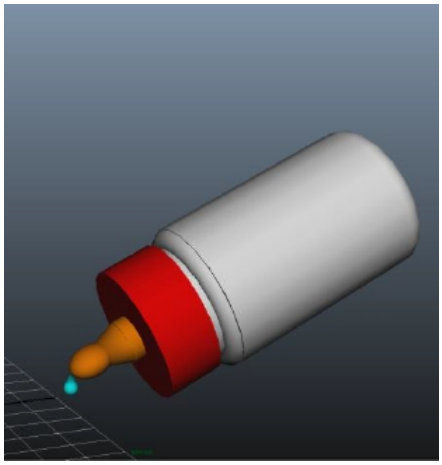


Abbildung 16: Animierte Flasche

Geschrumpft wird in Maya übrigens mithilfe der scale-Einstellung, welche als rot markierter Bereich in der nächsten Abbildung zu sehen ist.

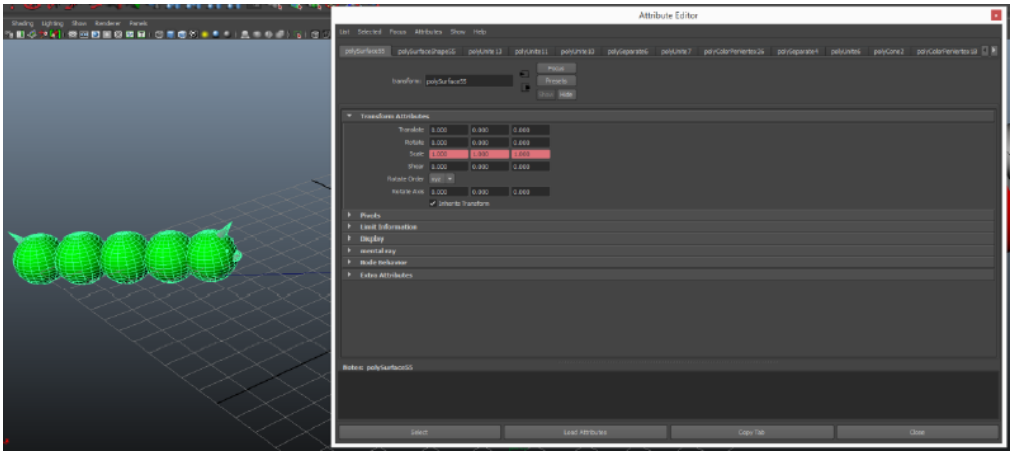


Abbildung 17: Schrumpfung der Raupe

Weiterhin wurde das bestehende Video „Bande" noch verbessert, indem die Bande insgesamt vergrößert wurde und eine Wiederholung des Aufschiebens der einzelnen Kugeln in Nahaufnahme eingefügt wurde.

5.7 7. Projektwoche

Zwar konnte unsere KI schon eine ganze Weile erfolgreich Tanken, aber KillerRaupi offiziell konnte es noch nicht, da noch kein Video dazu erstellt worden war. Das hat allerdings diese Woche glücklicher Weise ein Ende gefunden, denn nun kann KillerRaupi „tanken". In diesem Video fährt unsere Raupe auf eine Trinkflasche zu, die dann über Raupi hinwegschwebt, dabei kleiner, aber Raupi natürlich größer wird. Wie alle anderen Videos wurde auch dieses mit Windows Movie Maker zusammengefügt. Die Szene als KillerRaupi gerade die Flasche erreicht und sich Vollzusaugen beginnt ist in der nebenstehenden Abbildung zu sehen.

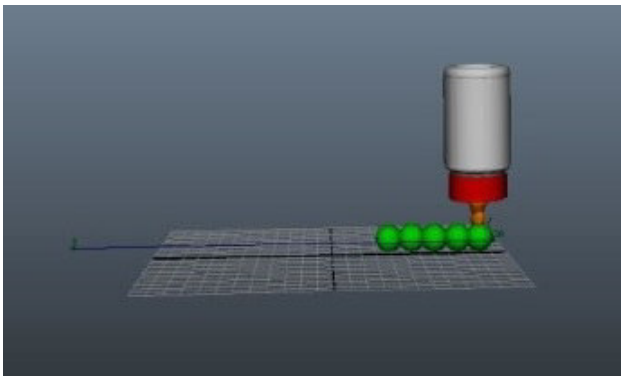


Abbildung 18: Szene aus dem Tankprozess

Des Weiteren wurde das bestehende Video „Bande" noch weiter verbessert. Die Kamera wurde näher an das Geschehen herangeholt und die einzelnen Körperkugeln schieben sich nun nicht mehr ineinander auf, sondern jede Kugel wird einzeln an der Wand „zerquetscht" und es bleibt nur eine Scheibe übrig. Die folgenden drei Abbildungen zeigen diesen Ablauf in Teilschritten.

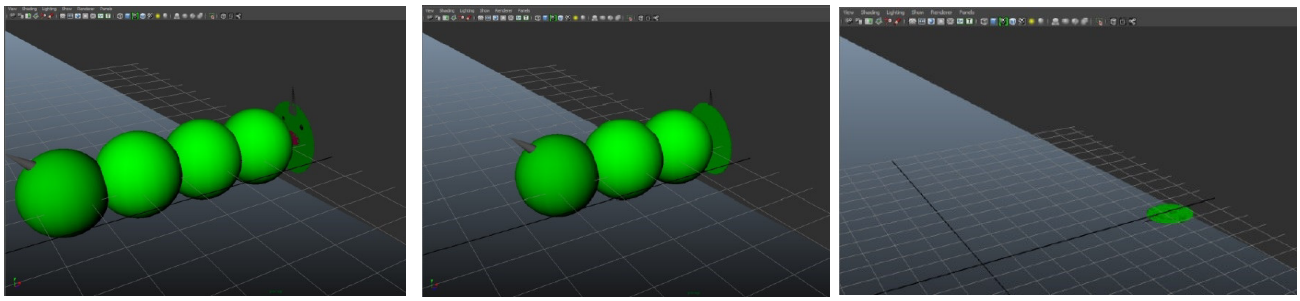


Abbildung 19: Neue Szenen des Videos „Bande"

Weiterhin hat das Animationsteam fleißig am Logo unserer Gruppe gearbeitet. Schon lange haben wir gehofft und uns das Logo ausgemalt, unter dem wir hoffentlich alle anderen Gruppen schlagen werden. Es war angedacht, dass die Initialen von KillerRaupi, also K und R, als Grundlage des Logos dienen und unsere Raupe sich durch die Buchstaben schlängelt. Glücklicherweise konnte das auch 1 zu 1 aus unserer Vorstellung in die Wirklichkeit übertragen werden. Noch ein wenig mit Perspektiven gespielt und die Beleuchtungsverhältnisse variiert ergaben sich so einige sehr schöne Logos, von denen jetzt nur noch das schönste ausgewählt werden muss.

Abbildung 20: Frische Logos

5.8 8. Projektwoche

Die Bearbeitung des Videos „Bande“ läuft ja bereits einige Wochen und wurde auch in den vergangenen sieben Tagen noch bearbeitet. Die Kameraperspektive wurde noch optimiert, indem während der „Fahrt“ der Raupe immer näher an diese heran gerückt wird, sodass ihr Ende an der Wand aus kürzester Distanz beobachtet werden kann. Das macht das Video in der Performance insgesamt harmonischer und interessanter.
Weiterhin haben wir befunden, dass die bisher aus 5 Kugeln bestehende Raupi nicht so schön aussieht, wie eine Raupe mit 6 Körperteilen, weshalb die alte kurze Raupe durch die längere in diesem Video noch ersetzt wurde. Das ist leider insofern recht zeitaufwendig, da der gesamte Kurzfilm noch einmal erzeugt werden muss.

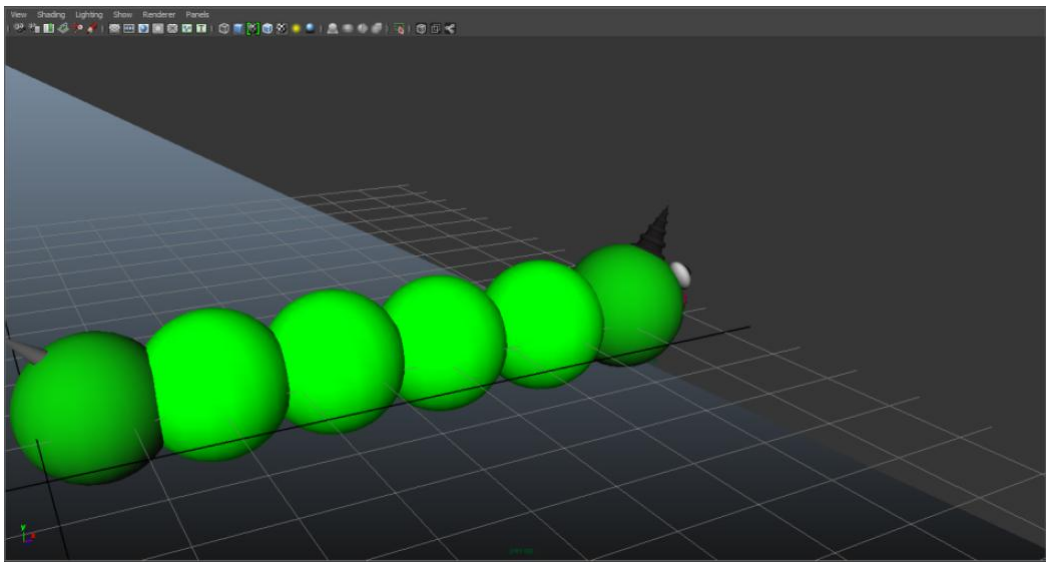


Abbildung 21: 6-teilige Raupe vor Bande

Des Weiteren wurde noch eine weitere Version des Videos „Bande“ erstellt. Dieses unterscheidet sich von dem anderen Film ausschließlich in der Kameraführung. Das Video startet mit einer Nahaufnahme und dreht sich während der Fahrt mit, sodass unsere KillerRaupi immer noch von Nahen zu sehen ist.
Auch dieses Video hat seinen Charme wir werden sehen, welches wir letztendlich auswählen werden.

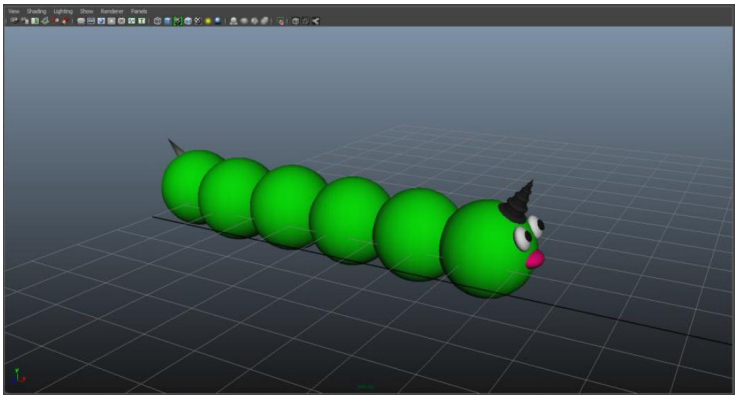


Abbildung 22: Nahaufnahme zu Beginn

Außerdem ist in dieser Woche noch ein ganz neues Video entstanden, von dem wir hoffen, dass es beim Turnier allerdings nie abgespielt werden muss. Wenn KillerRaupi

verliert, geht sie weinend nach Hause und knallt die Tür hinter sich zu, dass die Dachziegel nur so wackeln. Diese Video besteht aus den einzelnen Szenen, dass KillerRaupi auf ihr Haus zufährt, die Tür sich öffnet (ja, Raupi hat einen automatischen Türöffner) und sie rauscht durch die Tür, die sich schwungvoll schließt und alle Ziegel am Haus hüpfen energisch davon weg.
Dieses Video wurde vorerst mit der 5-teiligen Raupe erstellt. Weinen kann Raupi ebenfalls noch nicht, aber das wird definitiv in den nächsten Wochen angegangen werden.

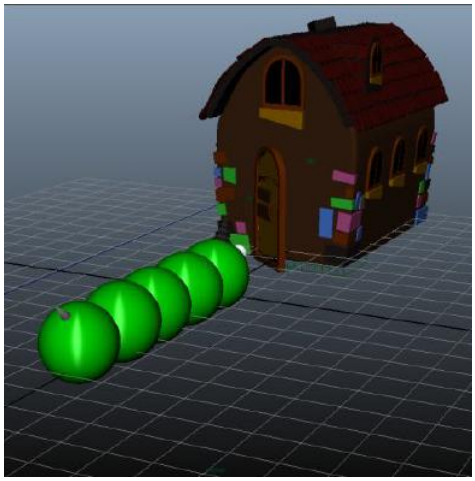


Abbildung 23: Raupi geht frustiert nach Hause

Des Weiteren wurde noch das Video „Mine" auf die 6-teilige Raupi umgestellt, sowie in Movie Maker einige Bilder aus den Szenen aussortiert. Diese Bilder haben dafür gesorgt, dass es so ausschaut als würde KillerRaupi beim Kontakt mit der Mine stehen bleiben. Der Ablauf Fahren-Kontakt-Explosion ist also nun vollständig reibungslos.

Außerdem wurden auf Bestellung noch einige schöne KillerRaupi-Szenen animiert, die auf der Website unter http://killerraupi.wix.com/killerraupi#!killerraupi-persoenlich/cfvg mit zu sehen sind. Diese sind auch untenstehend abgebildet.

Abbildung 24: Bilder für die Website

5.9 9. Projektwoche

In dieser Woche wurde in das Video „Verlieren" die 6-teilige Raupe noch eingefügt und außerdem weint KillerRaupi nun auf dem Weg zu ihrem Zuhause noch herzzerreißend, wie im nebenstehenden Bild zu sehen ist. Das Ende des Videos hat sich allerdings nicht geändert. Raupi fährt immer noch beleidigt in ihr Haus und wirft die Tür hinter sich zu.

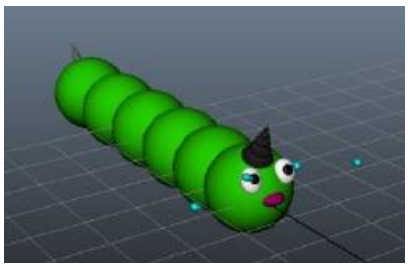
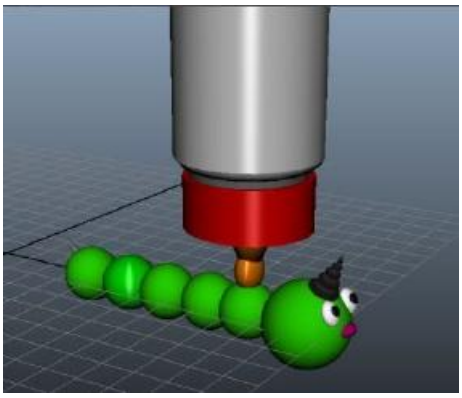


Abbildung 25: Weinende Raupi

Weiterhin wurde das Video „zu-wenig-Treibstoff" verbessert, indem die Trinkflasche nun nicht länger auf dem Boden liegt, sondern von der Decke herab hängt. Aus der Flasche tropft hierbei ständig eine Flüssigkeit und bildet eine Pfütze auf dem Boden.

Ebenso wurde auch das Video „Tanken" verbessert. Hier wird nun nicht mehr die gesamte Raupe „betankt", sondern eine Kugel nach der anderen, die dann entsprechend wächst.



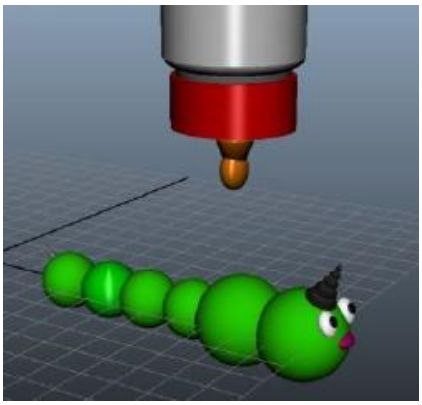


Abbildung 26: Tankvorgang

Diese Woche hat auch das schönste und glamouröseste Video, das wir hoffentlich ausschließlich sehen werden, begonnen zu entstehen. Das Video „Gewinnen" zeigt KillerRaupi wie sie (ähnlich wie im Video „Verlieren") nach Hause fährt und dann aber nicht die Tür zu knallt, sondern aus dem Schornstein als wunderhübscher Schmetterling aufsteigt und triumphierend davon fliegt.

Ein kleines Problem stellt hierbei allerdings noch der Übergang beim Aufsteigen aus dem Schornstein dar, also quasi das Entfalten der Flügel. Allerdings sind wir sehr zuversichtlich, dass diese Unstimmigkeit noch behoben werden kann. Ebenso müssen KillerRaupis Flügel natürlich noch schöne farbliche Nuancen erhalten, damit sie entsprechend strahlen kann.

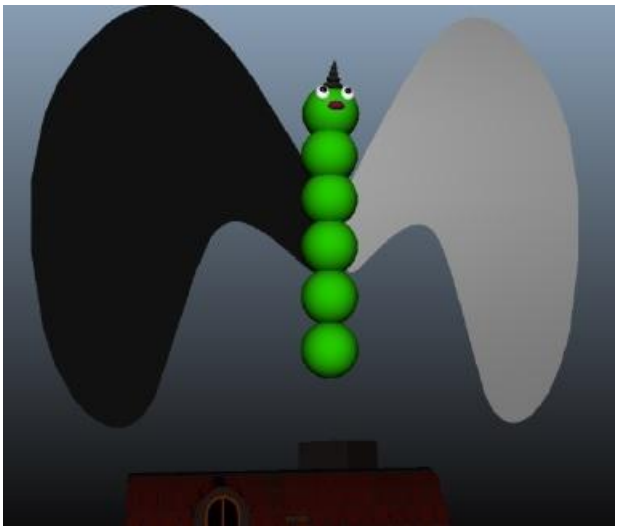


Abbildung 27: Raupi als Schmetterling

Des Weiteren wurde KillerRaupi diese Woche auch noch eine Raupenbewegung beigebracht. Hierbei schieben sich die Körperkugeln der Raupe zusammen und stretchen sich danach wieder, wobei trotzdem eine Vorwärtsbewegung der Raupe stattfindet. Das vermittelt tatsächlich den Eindruck einer kriechenden Raupe.

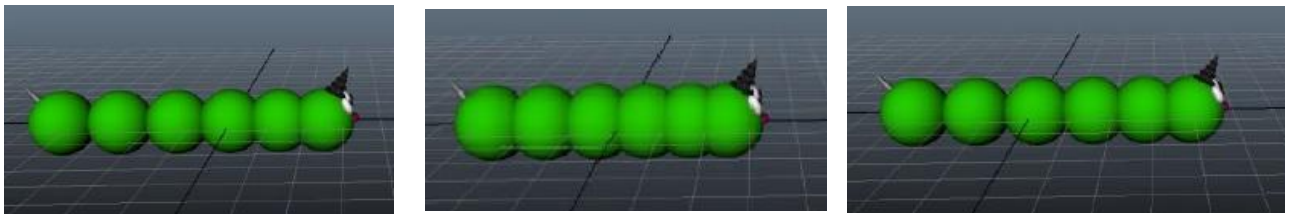


Abbildung 28: Animation der Raupenbewegung

5.10 10. Projektwoche

Recht häufig wurde in letzter Zeit unsere nicht so ganz schöne Mauer - die ja eigentlich auch gar keine ist, sondern eine dunkelgraue Wand - gegen die unsere KillerRaupi in dem Video „Bande" fährt, kritisiert. Deshalb musste endlich die Mauer weg! Oder besser gesagt eine schönere her.
Um das zu bewerkstelligen muss zuerst ein schlanker Quader in Form einer Mauer erstellt werden. Dieser muss dann mit dem gewünschten Material verkleidet werden, in unserem Fall mit einer soliden Steinmauer. Dies geschieht mithilfe des Attribute-Editors: Hier kann man als Material ein entsprechendes „file" importieren und gegebenenfalls noch rotieren oder anderweitig anpassen.

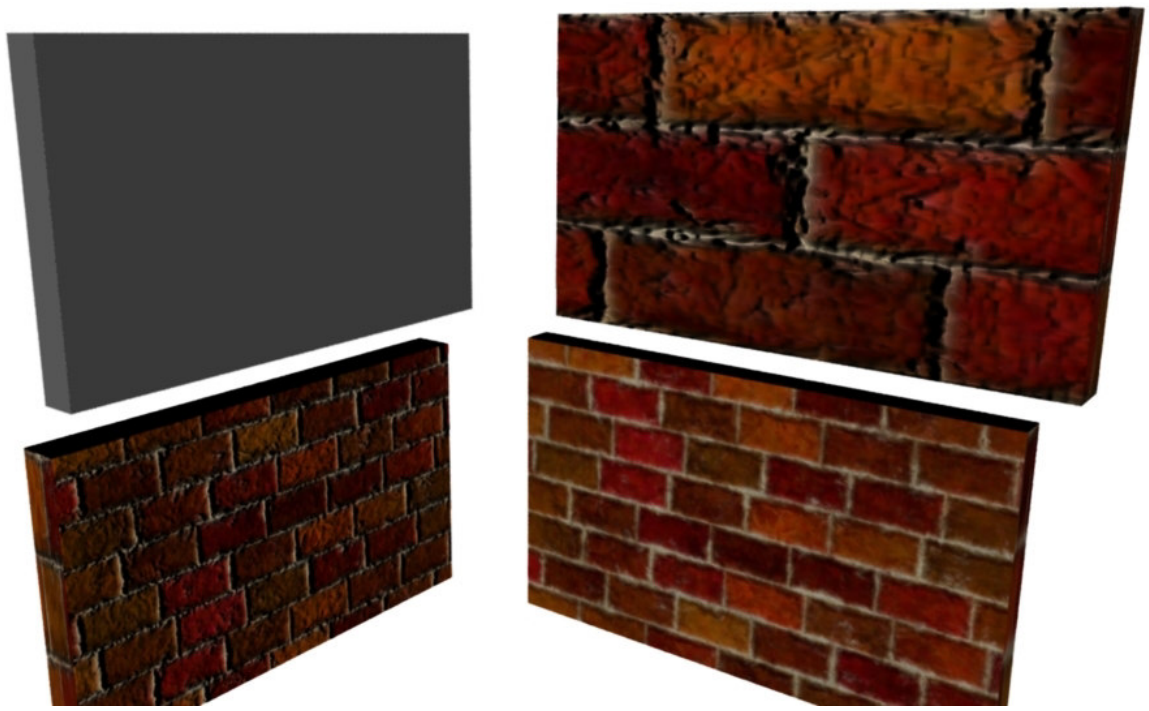


Abbildung 29: Rohmaterial der Mauern

Ohne zu rendern ist die Anpassung nur im „textured mode" sichtbar und nur 2D und leuchtet und reflektiert sehr stark. Um diese unerwünschte Nebenerscheinung zu beheben wird ein sogenanntes „planar mapping" auf die Backsteinmauer angewendet, welches die Textur neu projiziert. Weiterhin wurden die einzelnen Ziegel bzw. Backsteine skaliert, damit alles harmonisch aussieht.
Um schlussendlich eine hochwertige Textur inklusive Schattierungen zu erhalten, benötigt man eine gängige Technik des Virtual Designs: das „bump mapping". Hierbei handelt es sich um ein Tool um den Detailgrad von bestimmten Objekten zu verbessern, ohne die Komplexität in der Geometrie zu erhöhen. Die benötigten Informationen werden direkt in der Textur gelagert, mit deren Hilfe dann Schattierungen auf die Oberfläche gezeichnet werden können.

Nach diesem Schema wurden die verschiedenen Templates bearbeitet um unserem

Team eine gewisse Auswahl zu gewährleisten und eine schöne Mauer zu finden. Zum besseren Vergleich wurde vor dem Rendern aller Mauerstücke noch unsere Raupe in die Mitte importiert und entsprechend skaliert:

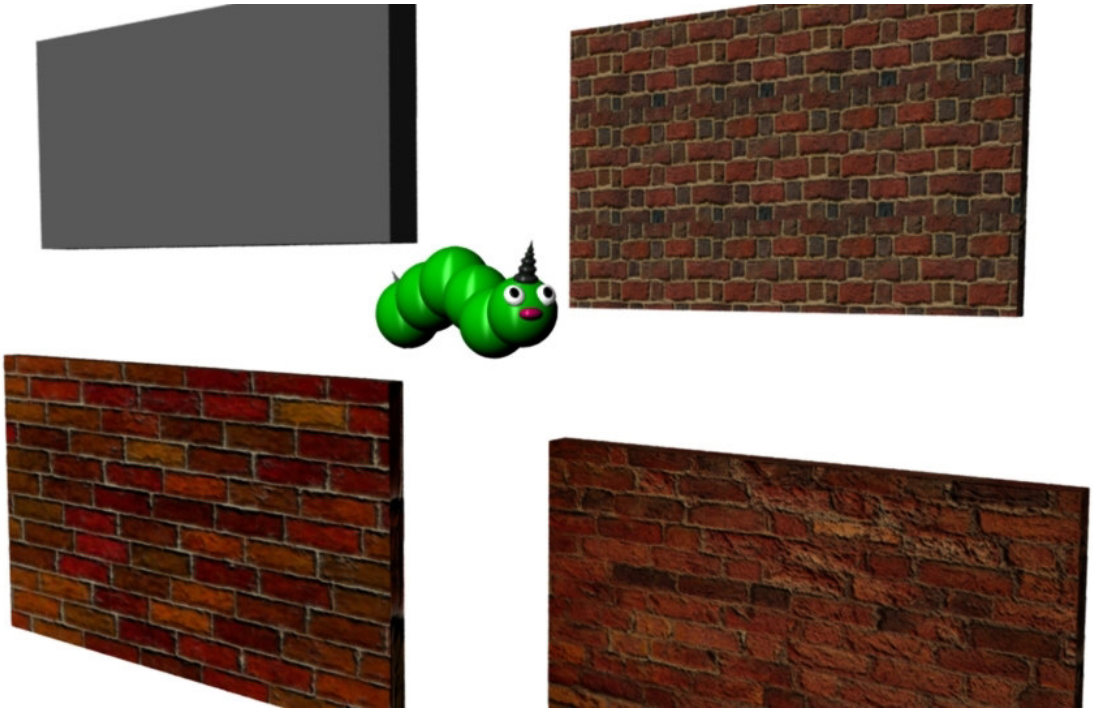


Abbildung 30: auswahlfertige Mauerstücke mit Raupi

5.11 11. Projektwoche

In der vergangenen Projektwoche wurde bereits begonnen die erarbeitete Raupenbewegung in die verschiedenen Videos einzufügen, beispielsweise in die Videos „Gewinnen" und „Mine", diese Woche konnte das auch noch für das Video „Bande" durchgeführt werden. In dieses Video wurde nun auch noch die in der letzten Woche erstellte und ausgewählte Mauer eingefügt, wie im nächsten Screenshot zu sehen ist.

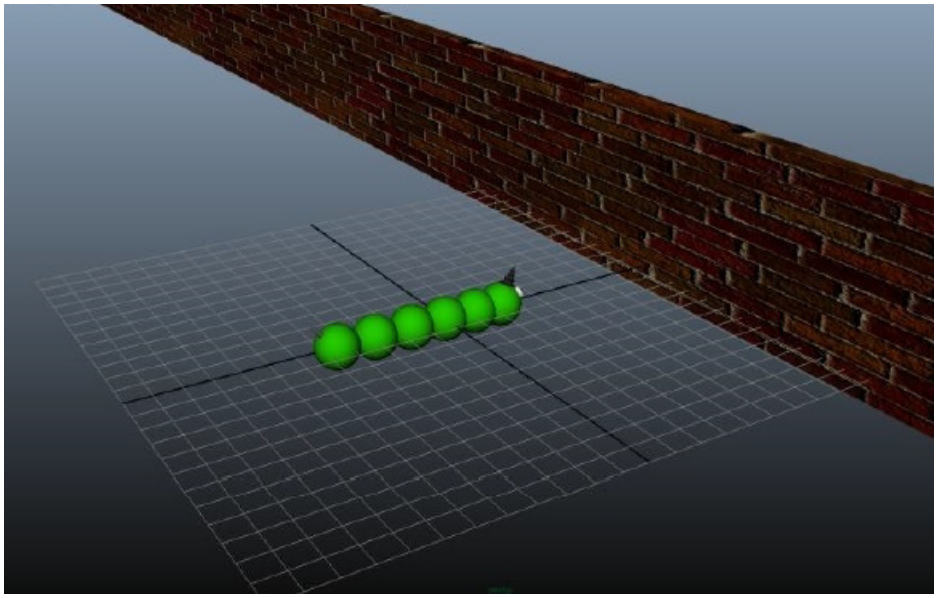


Abbildung 31: Video Bande mit neuer Mauer

Des Weiteren wurde die Raupenbewegung in dieser Projektwoche noch in das Video „Treibstoffmangel" eingefügt.

Das Einfügen der Raupenbewegung in dieses Video hatte allerdings zur Folge, dass sich das „Zerquetschen" an der Bande verändert. Schließlich bewegt sich die Raupe nun wesentlich langsamer auf die Mauer zu als noch bisher und daraus folgt, dass nun quasi jede einzelne Körperkugel an der Bande in relativ großen Zeitabständen zerquetscht wird.
Da das von uns allerdings nicht als negativ gewertet worden ist, haben wir hier nicht noch einmal Veränderungen dahingehend vorgenommen.

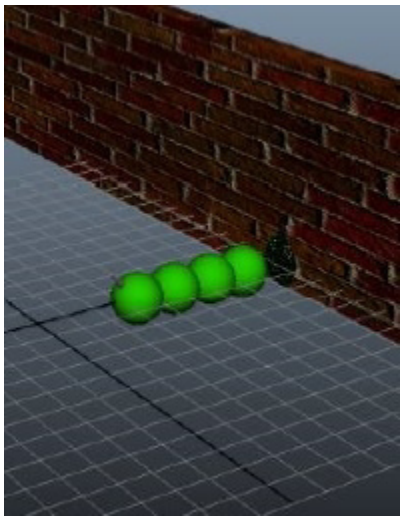
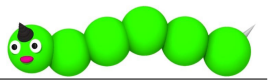


Abbildung 32: Zerquetschen



Des Weiteren ist uns aufgefallen, dass unsere Videos noch ein wenig unromantisch erscheinen, da sie immer nur vor einem tristen schwarzen Hintergrund stattfinden. Zu diesem Zweck wurde eine Spaceball-artige Umgebung von der Animation geschaffen. Diese zeigt einen erdachten Weltraum mit verschiedenen Planeten und vielen Sternen. KillerRaupi selbst bewegt sich hierbei auf einer dunkelgelben, gerundeten Planetenoberfläche. Diese Umgebung wurde auch bereits versuchsweise in das Video „Mine" eingefügt und gefällt uns dort sehr gut. In der nächsten Abbildung ist diese Umgebung zu sehen.

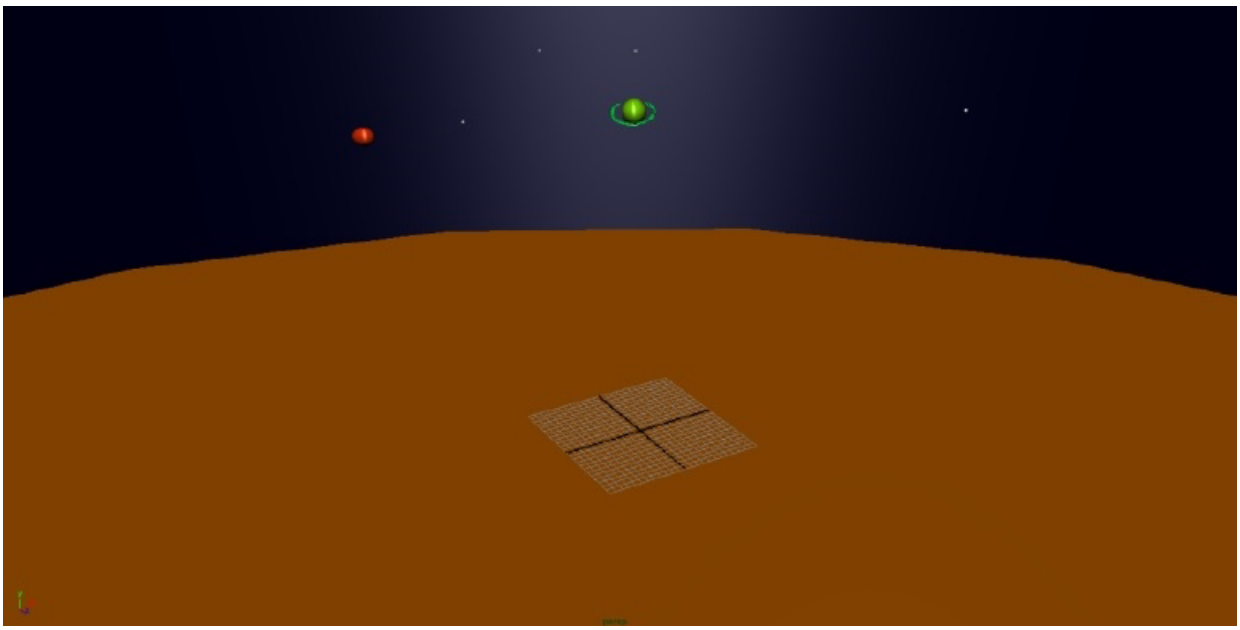


Abbildung 33: Spaceball-Umgebung

5.12 12. Projektwoche

In dieser Woche stand wieder einmal die Verbesserung und Komplettierung unserer bestehenden Videos im Mittelpunkt.
So wurde die Raupenbewegung noch in das Video „Tanken" und in das Video „Verlieren" eingefügt. In dem letztgenannten wurden ebenso KillerRaupis Tränen noch verbessert. Diese laufen nun nicht mehr in kleinen Tränchen aus ihren Augen, sondern als wahre Wasserfälle, die dann eine Pfütze auf dem Boden bilden.

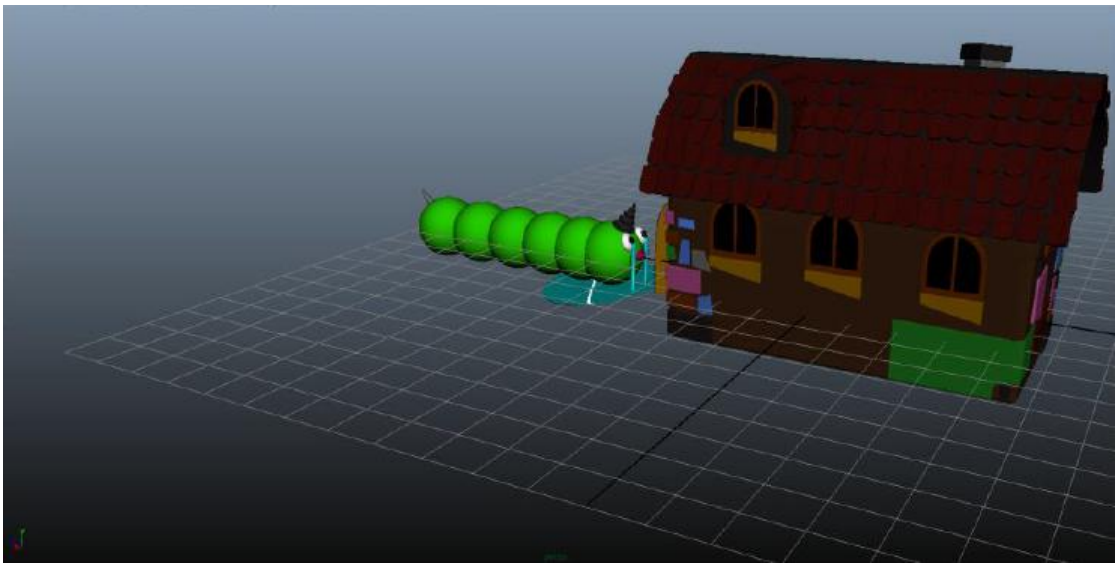


Abbildung 34: stark weinende KillerRaupi im Video „Verlieren"

Des Weiteren wurde die Spaceballs-Hintergrundumgebung in die Videos „Mine", „Bande" und „zu-wenig-Treibstoff" eingefügt. Damit ist dieser Arbeitsabschnitt auch noch für alle Clips geschafft.

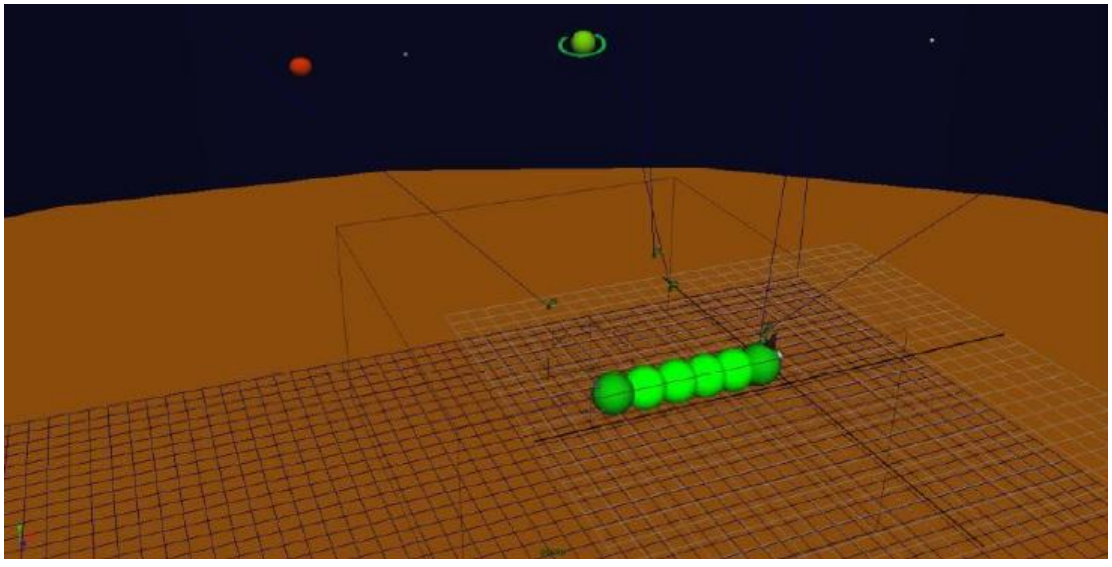
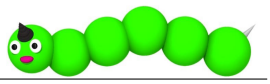
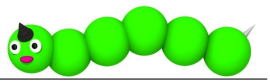


Abbildung 35: Arbeitsschritt KillerRaupi in den Hintergrund einzufügen



Außerdem konnte jetzt noch ein komplett neuer Arbeitsschritt begonnen werden, der bisher sehr zu kurz gekommen ist: der Sound.
Das Video „Mine" wurde hierbei als erstes mit entsprechenden Geräuschen unterlegt. Wenn man KillerRaupi einfach nur kriecht, hört man ein quietschendes Geräusch. Wenn Raupi auf die Mine trifft und dementsprechend eine Explosion erfolgt, so gibt auch der Sound einen Knall und eine verhallendes Grollen wieder.
Ebenso erfreut sich das Video „Bande" der neuen Geräusche. Wenn sich die Kugeln einzeln an der Mauer aufschieben, ist ein ploppender Ton zu hören. Wenn sich dann alle Kugeln aufgeschoben haben und die Scheiben alle von der Mauer auf den Boden klappen, hört man einen scheppernden Gong.
In der nächsten Projektwoche sollen noch so viele Videos wie möglich mit Ton versehen werden.



5.13 13. Projektwoche

Für diese vergangen Projektwoche gibt es nicht mehr viel zu sagen, da es sich um eine der letzten beiden Wochen handelt und unsere Designer der Animation sich ihre Arbeit sehr gut eingeteilt haben.
Als Feinschliff wurden nun noch in alle restlichen Clips passende Sounds eingefügt und somit die Arbeit an den Videos abgeschlossen.

Alle Videos sind auf unserer Homepage unter http://killerraupi.wix.com/killerraupi#!animation/caxr zu finden.

5.14 Zusammenfassung

Das Animationsteam hat mit Maya Autodesk zum einen Szenen mit KillerRaupi und Bestandteile aus ihrer Welt animiert, zum anderen auch die Clips „gedreht".
In allen ist die Hauptfigur „KillerRaupi" zu sehen. Eine grüne Raupe, bestehend aus 6 Körperkugeln, mit großen Glubschaugen, einem gezwirbelten Horn auf dem Kopf und einem Stachel auf ihrer letzten Körperkugel.

Eine der herausragendsten Animationen ist unser Logo (rechts zu sehen). Es umfasst die Initialen unserer KillerRaupi und die Raupe selbst ist ebenso mit verewigt, wie sie sich durch die Buchstaben schlängelt.
Eine Sonderanfertigung war die kleine Raupe, die in der Kopfzeile dieser Dokumentation von rechts auf der Kopflinie auf die Seite kriecht.
Eine besondere Anfertigung stellt das Haus unserer Raupe dar, ebenso wie einige Bilder, die unsere Raupe mit ihre Häuschen zeigen und auf der Website zu sehen sind.

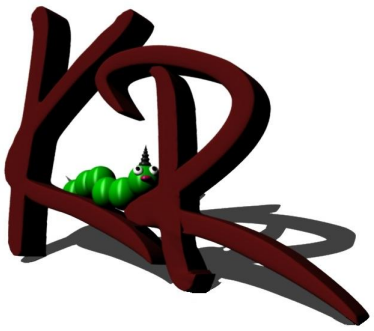
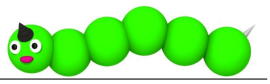


Abbildung 36: finales Logo

Das Herzstück der Arbeit der Animation sind allerdings natürlich die Videos[3] Alle besitzen als Hintergrund in den Szenen einen improvisierten Weltraum mit bunten Planeten und kleinen Sternen, sowie einen braunen Boden, über den KillerRaupi kriecht. Insgesamt wurden sechs Videos erstellt, alle besitzen auch eine entsprechende Tonuntermalung.
Das Video „Mine" wird, wie der Name schon sagt, eingeblendet, wenn unsere KillerRaupi in eine Mine auf dem Spielfeld hineinfährt. Das Video zeigt den gleichen Sachverhalt, gefolgt von einer großen Explosion, die alle Körperkugeln von Raupi in verschiedene Richtungen schleudert.
Der Clip „Bande" erscheint, wenn unserer Raupe gegen eine Bande gefahren ist. Im Video sieht man wie KillerRaupi auf eine eigens entworfene rote Backsteinmauer zufährt und sich beim Crash jede Kugeln einzeln an der Mauer plattdrückt und schlussendlich alle Raupischeiben auf den Boden klappen.
KillerRaupi muss um zu wachsen natürlich tanken und auch dazu gibt es ein Video. In diesem kriecht die Raupe zu einer eigens entworfenen Trinkflasche, die Raupi „betankt", indem jede Körperkugel berührt und dabei vergrößert wird.
Der gegenteilige Zustand, wenn die Tankstelle nicht mehr erreicht wird, ist in dem Video „zu-wenig-Treibstoff" festgehalten. Hier sieht man die immer stärker schrumpfende Raupe auf die Trinkflasche zufahren, die sie jedoch nicht mehr erreicht, weil sie schon vorher infinitesimal klein wird und verschwindet.
Sollte der unwahrscheinliche Fall auftreten, dass unsere KI doch einmal den Kürzeren zieht und dem Gegner den Sieg lassen muss, so zieht sich Raupi im Clip „Verlieren" heulend und schluchzend in ihre eigenen vier Wände zurück und knallt die Tür hinter sich zu, dass das Dach sogar ein Stück abhebt.

[3]Dies ist der Link zu unserer Homepage, wo man alle Clips anschauen kann: http://killerraupi.wix.com/killerraupi\protect\kern-.1667em\relaxanimation/caxr



Im Optimalfall jedoch wird unsere KI natürlich gewinnen und KillerRaupi kriecht im zugehörigen Video „Gewinnen" zwar ebenfalls in ihr Haus zurück, allerdings um dann als hübscher Schmetterling aus dem Kamin aufzusteigen.
Schlussendlich ist noch unsere 10s Intro-Musik zu erwähnen, die die Vorstellung unseres Teams untermalen wird.

An unseren Videos gefällt uns besonders gut, dass sie alle den gleichen Hintergrund und Stil besitzen, der sich als roter Faden durch diese Arbeit zieht. Mit der Weltraumumgebung ist es uns auch gelungen eine Brücke zu dem übergeordneten Thema „Spaceballs" zu schlagen und somit den Vorgaben des Projektes treu zu bleiben.

6 Fortschritte der Website

6.1 1. Projektwoche

An erster Stelle für das Designen der Website stand natürlich einen geeigneten Anbieter zu finden, bei dem das kostenlose Erstellen einer Website in HTML5 möglich ist und der eine anfängerfreundliche Oberfläche zum Kreieren der Website besitzt. Der betreuende Professor empfahl uns „wix.com" und nach ein wenig eigener Recherche fanden wir heraus, dass wix.com durchgehend gute Bewertungen erhalten hatte und alle anderen Kriterien ebenfalls zu erfüllen schien.
Hinter der Firma „wix.com" versteckt sich ein Homepage-Editor, der einen sehr starken und umfangreichen Baukasten zum kreativen Designen von Websites anbietet. Der Editor arbeitet mit einer optischen Bedienoberflächen, was eine Bedienung für ungeübte „Programmierer" enorm vereinfacht und gleichzeitig eine große Bandbreite an Möglichkeiten des Designs bietet.
Als erster Schritt musste nach optischen Gefallen ein sog. „Template", also eine Vorlage der Homepage, ausgewählt werden, dabei entschieden wir uns für das Design, welches in der nachfolgenden Abbildung zu sehen ist.

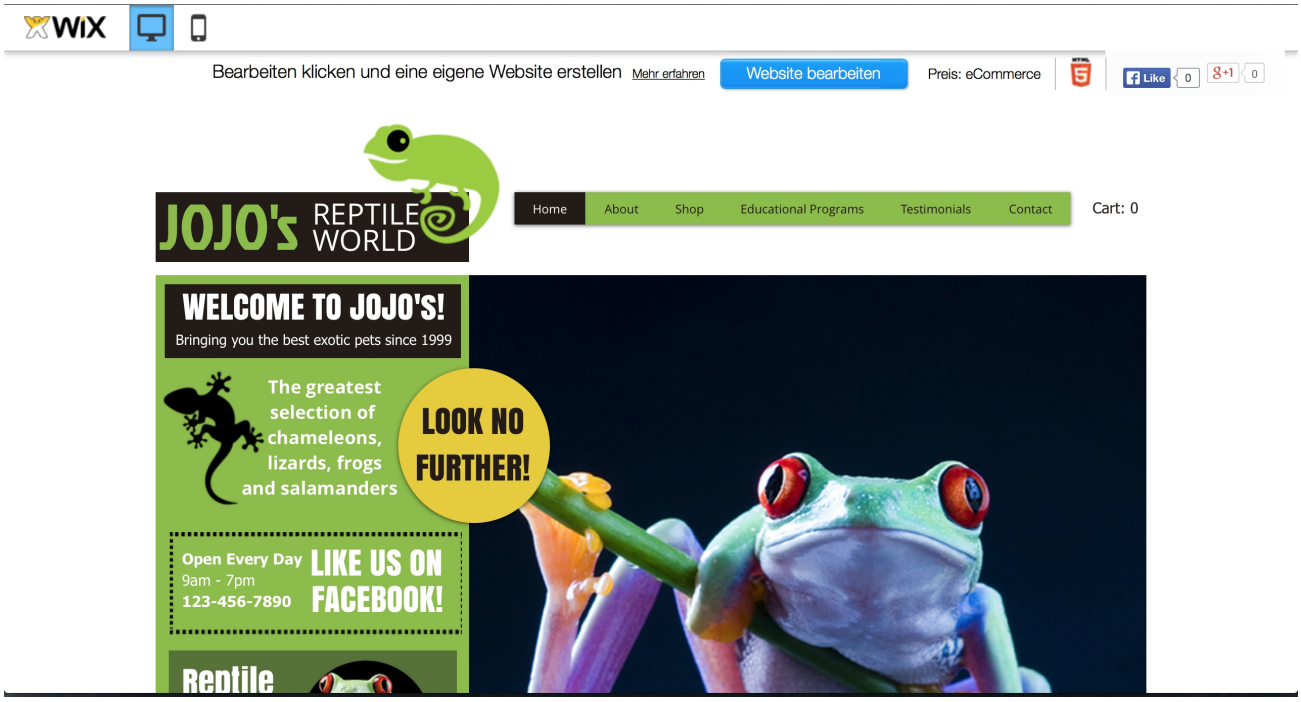
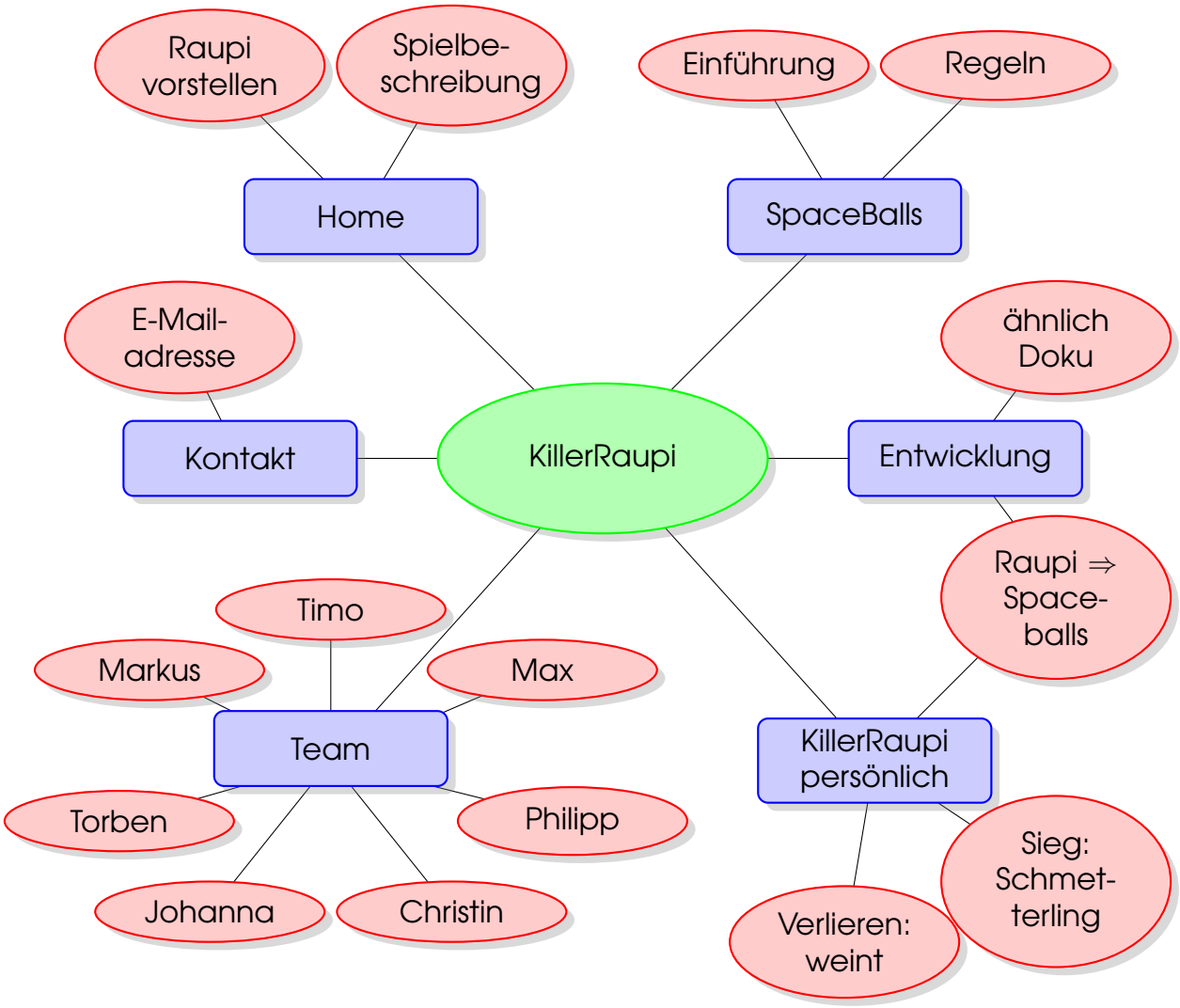


Abbildung 37: Vorlage für die Homepage

Unsere Wahl viel auf diese Vorlage, da das Design der Homepage sehr verspielt und niedlich ist, was man ja grundsätzlich auch mit einer Raupe, im besonderen unserer KillerRaupi, assoziieren sollte.

Des Weiteren mussten wir uns natürlich Gedanken darüber machen, welche Inhalte eigentlich auf der Website vorhanden sein sollten. Zu diesem Zweck wurde folgen-

den Mindmap erstellt:



Für die nächste Woche ist geplant, die gesammelten Inhalte der Mindmap in die Homepage zu übertragen und so eine erste Gliederung in die Website einzufügen und diverse Unterseiten zur weiteren Bearbeitung vorzubereiten.

6.2 2. Projektwoche

Nachdem in der letzten Woche die vorläufige Gliederung für die Website aufgestellt wurde, ging es nun daran diese in die Homepage einzupassen. Zuerst einmal wurden so die Überschriften für die Unterseiten in die Website mithilfe verschiedener Tools eingespeist. Das Resultat daraus zeigt die folgende Abbildung.

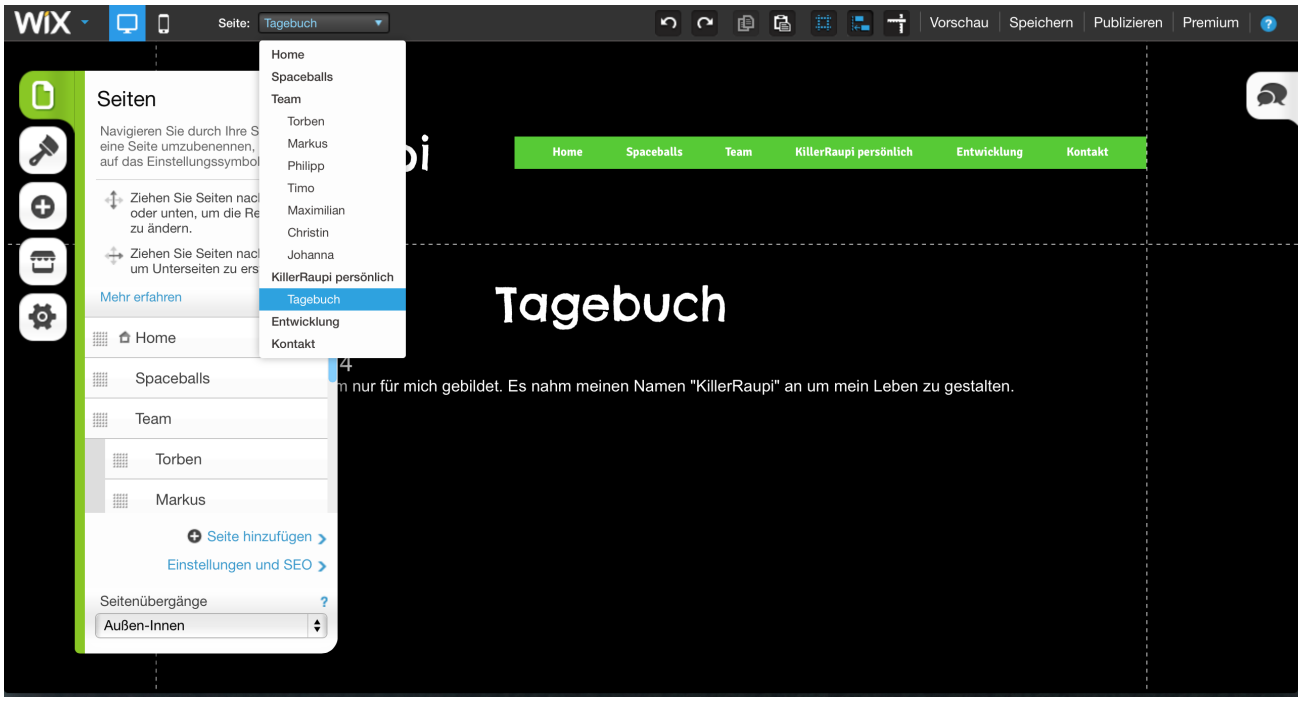


Abbildung 38: Erster Schritt zu den Unterseiten

Des Weiteren sollte diese Woche auch die Startseite zunehmend Gestalt annehmen. Ein sehr wichtiges Kriterium für die Startseite ist für uns eine sehr ansprechende Optik. Weiterhin soll natürlich auch unser Maskottchen KillerRaupi einen zentralen Platz einnehmen. Das kann natürlich erst geschehen wenn KillerRaupi fertig animiert wurde, was in dieser Woche allerdings noch nicht der Fall war.
Außerdem soll direkt auf der Startseite bereits eine kurze Einführung in den Inhalt des Projektes gegeben und die Spaceballs erklärt werden. Ein Gruppenbild von dem Team KillerRaupi soll ebenfalls auf der Startseite erscheinen, jedoch muss auch diese noch aufgenommen werden. Idealerweise mit einer Raupe natürlich, aber hier liegt auch erst einmal nur Idee, jedoch noch kein konkreter Plan für die Umsetzung vor.
Mit der Umsetzung für die Startseite und den daran gekoppelten Ideen wurde allerdings schon diese Woche begonnen, wie in der folgenden Abbildung zu sehen ist. Die Links zu den Unterseiten erscheinen in der oberen Leiste auf der Startseite. Eine ansprechende Überschrift ist ebenfalls bereits vorhanden.

Nächste Woche soll die Erstellung der Startseite weiter voran gebracht, sowie Texte für die Unterseiten geschrieben werden.

Abbildung 39: Anfänge beim Erstellen der Startseite

6.3 3. Projektwoche

Nachdem die Animatoren gegen Ende letzter Woche Animationen von KillerRaupi liefern konnten, war es nun möglich diese auf der Website zu verarbeiten. So wurde das erste Bild von Raupi auf der Homepage eingesetzt, und auch eine Animation mithilfe des Website-Editors ausprobiert. Diese sorgt dafür, dass die Raupe oben links in der Ecke, die auch auf jeder Unterseite zu sehen ist, beim ersten Aufruf der Seite ins Bild „schwebt". Das Resultat sieht also so aus:

Abbildung 40: Homepage mit KillerRaupi

Weiterhin wurden die Bilder, die KillerRaupi in verschiedenen Posen zeigen, bearbeitet und Accessoires hinzugefügt. Auf diese Weise haben wir eine kleine Bildergalerie erstellt, die auf der Seite „KillerRaupi persönlich" angelegt worden ist. Diese Unterseite ist in der nächsten Abbildung zu sehen:

Abbildung 41: Unterseite Bildergalerie der Website

Abbildung 42: Kontaktseite

Des Weiteren haben wir eine Kontaktseite erstellt, die in der rechten Abbildung zu sehen ist. Mithilfe dieser Kontaktseite kann direkt eine Mail an KillerRaupis E-Mail-Adresse verschickt werden. Diesen Account hatten wir bereits zu Beginn des Projektes eingerichtet. Das stellte anfangs ein kleines Problem dar, dass KillerRaupi ja 0 Jahre alt war, aber bei Google-Mail wohl niemand im Alter von 0 Jahren eine E-Mail-Adresse halten darf und wir so einige Probleme hatten und wir KillerRaupi am Ende älter machen mussten, als sie tatsächlich ist.
Außerdem haben wir damit begonnen eine Website-Version zu entwerfen, die Smartphone-tauglich ist. Diese soll in den nächsten Wochen noch verbesserte werden. Erste Erfolge sind in der nächsten Abbildung zu sehen:

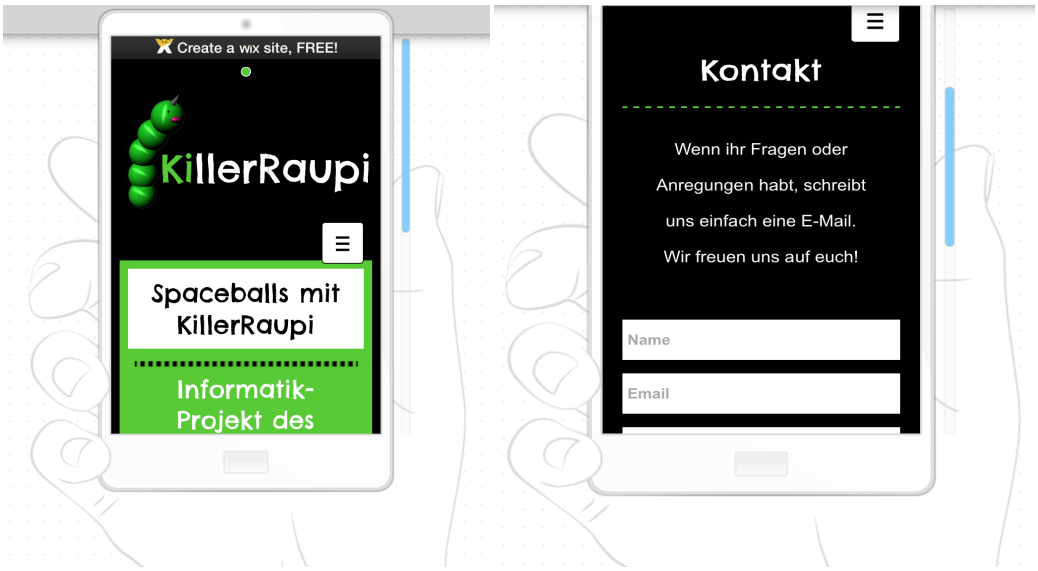


Abbildung 43: Erste Versionen für das Smartphone

6.4 4. Projektwoche

Da in letzter Zeit ein wenig in Kritik stand, dass unsere Website Gefahr läuft zu kindlich zu werden, haben wir uns entschieden einen sachlichen Steckbrief über Raupen allgemein zu erstellen, der allerdings auf KillerRaupi zugeschnitten ist. Das vorläufige Ergebnis ist in der folgenden Abbildung zu sehen:

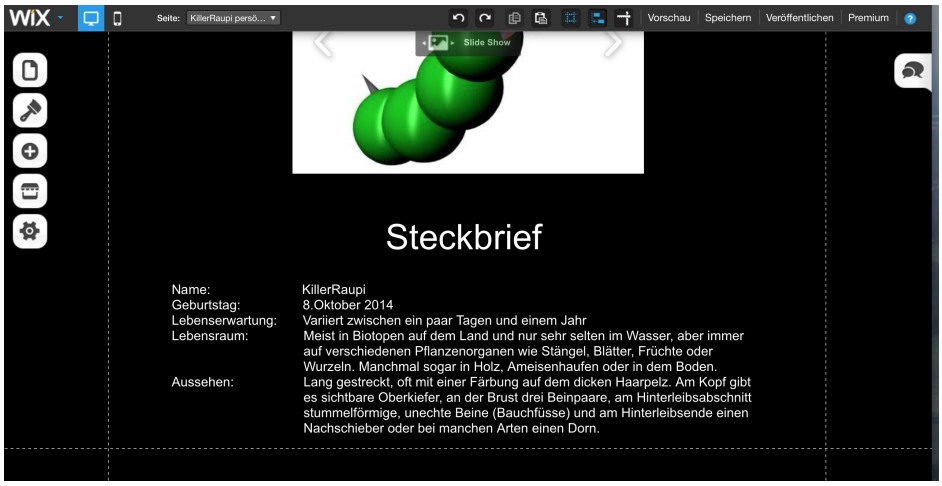


Abbildung 44: Erste Fassung des Steckbriefs

Diese Version wurde von unserer Editorin für allerdings noch nicht zufrieden stellend befunden und wird deshalb noch in nächster Zeit überarbeitet werden müssen.
Des Weiteren wurde noch ein Impressum erstellt. Zu diesem Zweck haben wir auch recherchiert, welche rechtlichen Vorgaben ein Impressum erfüllen muss und dementsprechend eine erste Version erarbeitet, welche in der nächsten Abbildung zu sehen ist.

Abbildung 45: Erste Fassung des Impressums

Auch hier gibt es allerdings noch Überarbeitungsbedarf und offene Fragen bezüglich der Rechte, was in den nächsten Tagen und Wochen noch erledigt werden sollte.

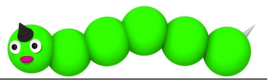
6.5 5. Projektwoche

Diese Projektwoche war eine recht schwarze Woche für unsere Website. Es sind leider einige Probleme aufgetaucht, die einer Lösung bedurften und deshalb diverse Umstrukturierungsmaßnahmen in der Gliederung der Website nötig machten. Uns fiel auf, dass man bei dem Menüpunkt „Team" nur die ersten beiden Unterseiten anwählen könnte. Die darauf folgenden waren nicht zu öffnen. Der Grund dafür war, dass der Website-Baukasten „wix.com" nur maximal zwei Unterseiten zulässt[4]. Weitere Seiten können zwar gestaltet werden, jedoch nicht im Vorschau-Modus geöffnet werden.
Auf der Hilfeseite von wix.com und in diversen Foren war dieses Problem zwar nicht beschrieben, allerdings wurde diese Theorie auch durch zahlreiche Beispiel-Websites von wix.com bestätigt, da diese ebenfalls maximal zwei Unterseiten besaßen.
Aus diesem Grund mussten die Unterseiten neu sortiert werden, wie in den nächsten Abbildungen zu sehen ist:

Abbildung 46: Neue Gliederung der Unterseiten

Die kompletten Unterseiten jedes einzelnen Teammitgliedes fielen weg, deren Inhalt wird auf der Seite „Team" zu finden sein.
Für die entstehenden Videos und Spielaufzeichnungen wurden zwei Unterseiten in dem Menüpunkt „Spaceballs" hinzugefügt.

[4] Wie durch ein Wunder war es später doch möglich mehrere Unterseiten zu erstellen, siehe Projektwoche 10 (Seite 107)



Bei den Videos allerdings ergab sich ein neues Problem. Auf die Homepage direkt können keine Videos hochgeladen, sondern nur Links zu den Videos eingefügt werden. Man könnte sie zwar mit html einfügen, aber nur mit ca. 8000 Zeichen, das ist für uns allerdings nicht ausreichend, weshalb Videos also tatsächlich nicht hochgeladen werden können. Eine Alternative könnte darin bestehen die Videos vom Bildschirm mitzuschneiden oder alternative Aufnahmemethoden zu verwenden und diese dann auf externe Plattformen, bspw. Youtube, hochzuladen und dann zu verlinken.

6.6 6.Projektwoche

Im Laufe dieser Woche wurde das Herz der Website erstellt: eine ausführliche Beschreibung des Informatik-Projektes und natürlich des Spiels „Spaceballs" im Detail. Ein Auszug dieser Seite ist auch in der folgenden Abbildung zu sehen:

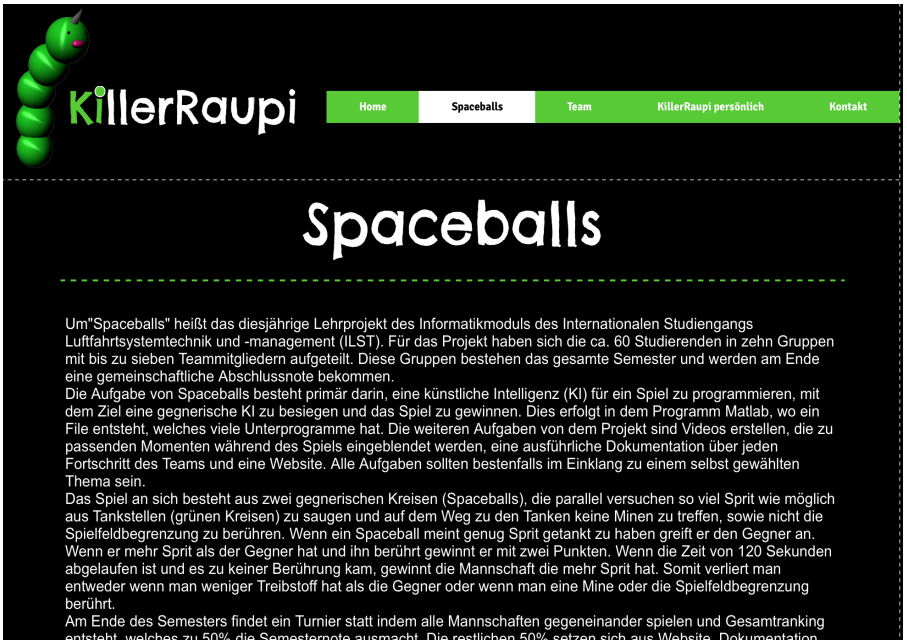


Abbildung 47: Auszug aus der Seite „Spaceballs"

Weiterhin beinhaltet dieser Teil eine kurze Zusammenfassung unserer Umsetzung der Aufgaben und Pläne im Projekt. Am Ende dieser Zusammenfassung gibt es außerdem einen Link (in der folgenden Abbildung grün hervorgehoben), der den Besucher der Website zu der Seite „KillerRaupi persönlich" zum Weiterlesen führt.

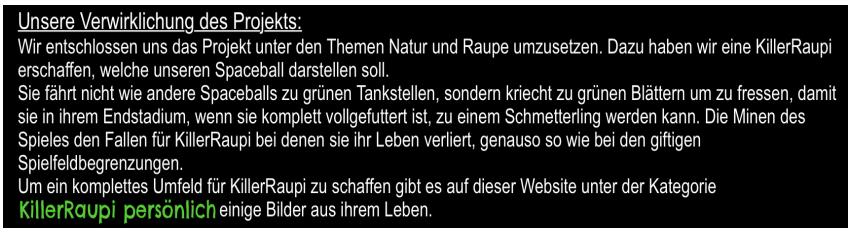


Abbildung 48: Zusammenfassung der Aufgaben mit Link

Eine weitere unbeantwortete Frage bleibt vorerst, wie rechtlich-korrekt das Impressum sein sollte. Von dem betreuenden Professor haben wir die Anweisung erhalten ein rechtlich gültiges Impressum zu entwerfen. Dies beinhaltet allerdings private Daten der Administratoren, die wir in diesem Umfang nicht in das World Wide Web einspeisen möchten und sich damit die Frage in den Raum drängte, ob man die Kontaktdaten der Hochschule zu diesem Zwecke nutzen könnte. Leider blieb unsere Anfrage an das Rechenzentrum bis heute vorerst unbeantwortet und somit das Impressum unvollständig.

6.7 7.Projektwoche

Wie in der letzten Woche begonnen wurde KillerRaupis Tagebuch auf den heutigen Stand der Programmierung gebracht. Dieses Tagebuch beinhaltet aus KillerRaupis Sicht die Stadien ihrer Entwicklung. Dass sie beispielsweise lernt zu fressen (tanken) oder nicht mehr gegen Wände (die Bande) kriecht usw.

Abbildung 49: KillerRaupis Tagebuch

Weiterhin wurde die Unterseite eingeteilt, auf welcher einmal unsere Gruppe vorgestellt werden soll. Zuoberst prangt ein Gruppenbild (derzeit noch ein Platzhalter) und im weiteren Verlauf schließen sich die Namen der Mitglieder (Ladies first) mit einem Porträtfoto (derzeit ebenfalls noch ein Platzhalter) und einer kurzen Beschreibung des Aufgabenfeldes an. Die zugehörige Abbildung ist auf der nächsten Seite zu finden.
Hier müssen natürlich in nächster Zeit noch die tatsächlichen Fotos eingefügt werden und der entsprechende Text geschrieben werden, aber die Struktur steht soweit erst einmal fest.
Zu überlegen ist auf jeden Fall, ob Dokumentation und Website in diesem Punkt nicht zusammen arbeiten könnten, da die Texte ja in beiden Medien erscheinen werden.

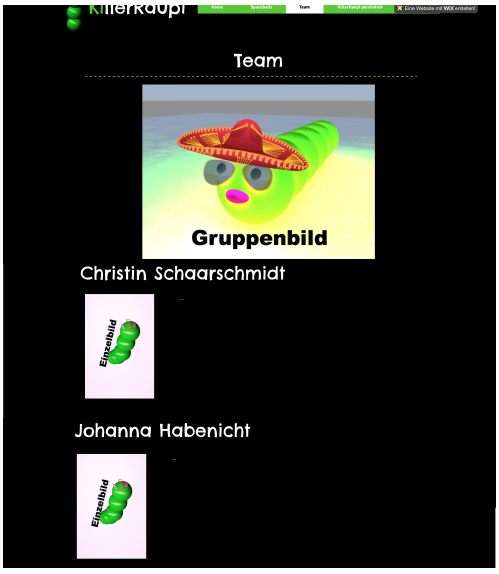


Abbildung 50: Unterseite „Team"

Weiterhin hat diese Woche das Rechenzentrum der Hochschule geantwortet. Die Antwort war derartig formuliert, dass eine Website nur dann die Daten der Hochschule anführen darf, wenn es sich um eine Website der Hochschule handelt. Nett gesagt, leider bringt uns das in unserem Problem bis jetzt kein Stück weiter und das Impressum muss deswegen wohl oder übel noch unvollständig bleiben.

6.8 8.Projektwoche

Auf der Startseite wurde nun eine der fundamentalsten Änderungen vorgenommen: Unser wundervolles, noch gar nicht so lange bestehendes Gruppenlogo kann nun seinen vorgesehen Platz in der linken oberen Ecke der Startseite einnehmen. Wie bereits das Platzhalterbild zuvor schwebt unser Logo beim ersten Öffnen der Seite herein und bleibt dann beständig zu sehen.

Abbildung 51: Startseite mit Logo

Des Weiteren wurde die Unterseite, die unser Team vorstellt, mit Inhalt gefüllt und enthält nun eine kurze Beschreibung zu jedem Mitglied. Die Porträts existieren noch nicht, daher werden sie erst einmal noch von Platzhaltern besetzt. Zwei Beschreibungen sind untenstehend exemplarisch abgebildet.

Abbildung 52: Unterseite Team mit Inhalt

Ebenso hat KillerRaupi diese Woche ihren lang ersehnten Steckbrief bekommen, in dem mehr von ihrer amüsanten Hintergrundgeschichte erfährt. Diese Seite soll die strenge Hochschulprojekt-Geschichte ein wenig auflockern.

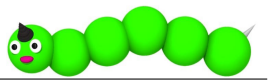
6.9 9. Projektwoche

Wie in der Dokumentation der Animation in den letzten Wochen zu lesen war, hatte KillerRaupi vor einiger Zeit ihr Aussehen verändert und runde Glubschaugen bekommen. Folglich waren alle Bilder in der Galerie unserer Website veraltet und wurden diese Woche durch aktuelle ausgetauscht. Ebenso erhielt jedes Bild einen Untertitel und eine kurze Bildbeschreibung, die dem Besucher der Website einen Eindruck in die Fantasiewelt von Raupi geben soll. Einige Bilder wurden auch gelöscht, andere hingegen überarbeitet und vor allem neue, beispielsweise KillerRaupi in ihrem Haus, hinzugefügt. Eine Auswahl der Galeriebilder ist im Folgenden zu sehen.

Abbildung 53: Auswahl an Galeriebildern

In Zusammenarbeit mit dem Animationsduo wurde auch noch einmal ein verbessertes Logo entworfen, das nun direkt wieder auf die Startseite eingesetzt werden konnte.
Ebenso wurde die Unterseite spezielle Teammitglieder überarbeitet und nimmt nun auch zügig Form an. Diese Unterseite widmet sich Raupen, denen man (erstaunlich oft) im Alltag begegnet, wenn man darauf achtet. Dort wurden auch bereits Bilder mit eingefügt, allerdings fehlt an der einen oder anderen Stelle noch ein schöner Text, was in den nächsten Wochen natürlich noch ergänzt werden muss.
Des Weiteren wurde auch die Handyversion diese Woche aktualisiert und vor allem verbessert, indem die Schriftgröße variiert wurde und Text anders angeordnet wurde als in der normalen Desktop-Version. Einige Seiten davon sind in der nächsten Abbildung zu sehen.

Abbildung 54: Überarbeitete Seiten der Smartphone-Version



Weiterhin wurde die Entwicklung aus Sicht von KillerRaupi noch auf den neusten Stand gebracht, schließlich unterliegt da die Website ebenso den aktuellen Änderung der Programmierung, wie die Dokumentation.
Zuletzt kann für diese Woche noch verbucht werden, dass das Impressum nun endlich noch vervollständigt wurde.

6.10 10. Projektwoche

Diese Woche konnte endlich eine lange begonnene Arbeit fertiggestellt werden: die Unterseite „spezielle Teammitglieder". Diese Seite beinhaltet alle amüsanten Raupen-Besuche, die unser Team während der Projektzeit empfangen hat. Dazu gehört das singende Kinderspielzeug Rolli-Raupe, aber auch essbare Fruchtgummi-Raupen, die vom Team besonders herzlich angenommen wurden. Diese haben also auch Eingang auf der Website gefunden und werden mit einem Foto und einem kurzen Text vermerkt. Sollten wir noch mehr Raupen aus dem „alltäglichen" Gebrauch begrüßen dürfen, dann werden auch die natürlich noch auf der Website vermerkt werden.

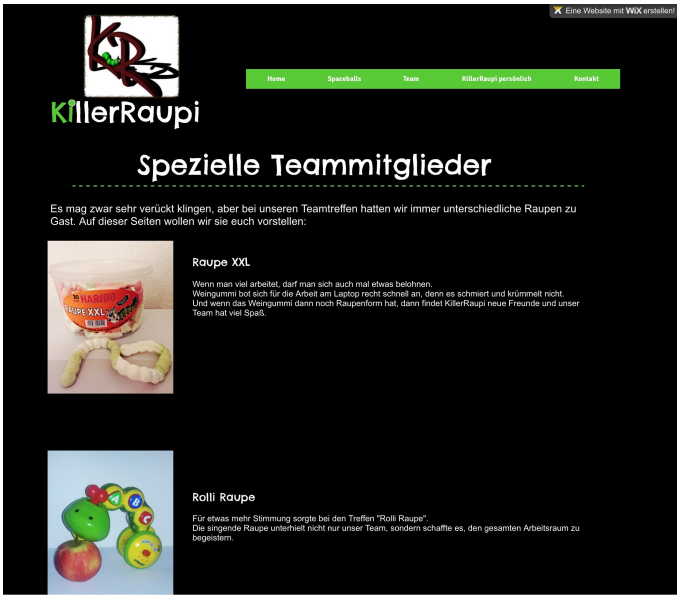
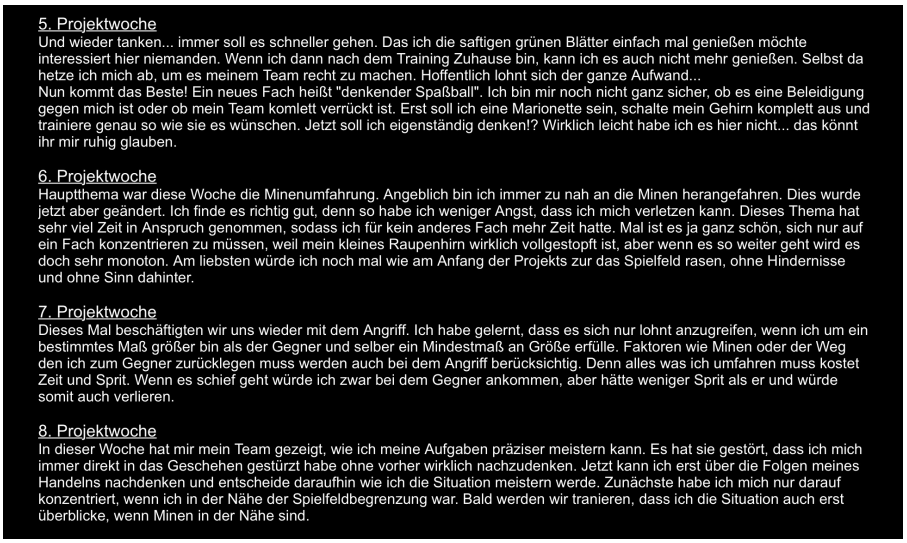


Abbildung 55: Unterseite „spezielle Teammitglieder"

Ein kleiner Strolch ist nach wie vor die Smartphone-Version, denn diese sieht es irgendwie nicht so ganz ein sich ebenso zu formatieren, wie die Desktop-Version. Deshalb bestand die Hauptarbeit dieser Woche darin die Mobilvariante noch manuell anzupassen, damit alles hübsch aussieht. Ebenso wurde in der Mobilversion ein Button eingefügt, der am rechten Rand mitläuft und den Besucher der Website immer zurück zum Seitenanfang bringt, wenn er das möchte.

Abbildung 56: aktualisierte Mobilversion

Ebenso wurde die Entwicklung von KillerRaupi noch auf den neusten Stand gebracht (anhand der Doku, wie bisher).



5. Projektwoche
Und wieder tanken... immer soll es schneller gehen. Das ich die saftigen grünen Blätter einfach mal genießen möchte interessiert hier niemanden. Wenn ich dann nach dem Training Zuhause bin, kann ich es auch nicht mehr genießen. Selbst da hetze ich mich ab, um es meinem Team recht zu machen. Hoffentlich lohnt sich der ganze Aufwand...
Nun kommt das Beste! Ein neues Fach heißt "denkender Spaßball". Ich bin mir noch nicht ganz sicher, ob es eine Beleidigung gegen mich ist oder ob mein Team komlett verrückt ist. Erst soll ich eine Marionette sein, schalte mein Gehirn komplett aus und trainiere genau so wie sie es wünschen. Jetzt soll ich eigenständig denken!? Wirklich leicht habe ich es hier nicht... das könnt ihr mir ruhig glauben.

6. Projektwoche
Hauptthema war diese Woche die Minenumfahrung. Angeblich bin ich immer zu nah an die Minen herangefahren. Dies wurde jetzt aber geändert. Ich finde es richtig gut, denn so habe ich weniger Angst, dass ich mich verletzen kann. Dieses Thema hat sehr viel Zeit in Anspruch genommen, sodass ich für kein anderes Fach mehr Zeit hatte. Mal ist es ja ganz schön, sich nur auf ein Fach konzentrieren zu müssen, weil mein kleines Raupenhirn wirklich vollgestopft ist, aber wenn es so weiter geht wird es doch sehr monoton. Am liebsten würde ich noch mal wie am Anfang der Projekts zur das Spielfeld rasen, ohne Hindernisse und ohne Sinn dahinter.

7. Projektwoche
Dieses Mal beschäftigten wir uns wieder mit dem Angriff. Ich habe gelernt, dass es sich nur lohnt anzugreifen, wenn ich um ein bestimmtes Maß größer bin als der Gegner und selber ein Mindestmaß an Größe erfülle. Faktoren wie Minen oder der Weg den ich zum Gegner zurücklegen muss werden auch bei dem Angriff berücksichtig. Denn alles was ich umfahren muss kostet Zeit und Sprit. Wenn es schief geht würde ich zwar bei dem Gegner ankommen, aber hätte weniger Sprit als er und würde somit auch verlieren.

8. Projektwoche
In dieser Woche hat mir mein Team gezeigt, wie ich meine Aufgaben präziser meistern kann. Es hat sie gestört, dass ich mich immer direkt in das Geschehen gestürzt habe ohne vorher wirklich nachzudenken. Jetzt kann ich erst über die Folgen meines Handelns nachdenken und entscheide daraufhin wie ich die Situation meistern werde. Zunächste habe ich mich nur darauf konzentriert, wenn ich in der Nähe der Spielfeldbegrenzung war. Bald werden wir tranieren, dass ich die Situation auch erst überblicke, wenn Minen in der Nähe sind.

Abbildung 57: aktualisiertes Tagebuch

Weiterhin ist noch eine neue Unterseite hinzugekommen: „Entwicklung der Animation". Da der Name ja bekanntlich Programm ist, enthält diese Seite natürlich die Fortschritte der Animation über alle Wochen. Allerdings steht hierfür erst einmal nur das Grundgerüst, aber das wird sich in der nächsten Zeit noch gut mit Inhalt füllen. Diese Unterseite findet man bei „Spaceballs". Der aufmerksame Dokumentationsleser stellt jetzt natürlich fest: noch eine Unterseite bei Spaceballs? Das geht doch gar nicht...! Doch, es geht seit jüngster Zeit. Wir sind uns nicht ganz sicher wie das geschehen konnte, ob wix etwas geändert hat oder wo nun der Hase im Pfeffer liegt, Fakt ist, man kann nun mehr als 2 Unterseiten bei einem Inhaltspunkt erstellen, auch der Vorschaumodus funktioniert einwandfrei. Wir werden uns jetzt nicht weiter fragen, was wir da wohl anfangs übersehen haben, sondern uns einfach freuen, dass es jetzt funktioniert und diese Funktion nutzen.

6.11 11. und 12. Projektwoche

Die letzten beiden Projektwochen (die über die Weihnachtsfeiertage verliefen) sind recht schwierig zu trennen, deswegen sollen sie hier in der Dokumentation auch gemeinsam erscheinen.
In der 11. Projektwoche war es uns ja endlich gelungen die lang angekündigten Porträtaufnahmen und das Gruppenbild zu schießen, was es nun zu bearbeiten galt. Jetzt wundert sich der erfahrene Dokumentationsleser natürlich, warum dieser Inhalt ausgerechnet in dem Kapitel „Website" auftaucht, da die Bildbearbeitung doch vermutlich eher in den Bereich „Animation" passt! Tatsächlich ist es so, dass wir bis heute noch keine Rückmeldung von den Editoren des Programmes, mit dem wir unsere Bilder cartoon-haft bearbeiten wollten, erhalten haben und folglich rechtlich nicht auf der abgesicherten Seite sind, sollten wir es doch verwenden. Deshalb hat sich unsere Website-Designerin beherzt dieses Problems angenommen und mit dem zugehörigen Programms eines bekannten Elektronikherstellers, der viel mit Obst zu tun hat, unsere Porträts bearbeitet. Das geschah im Detail mit dem Effekt „San Carmen", der die Lichtverhältnisse und Farben bzw. Eigenschaften des Fotos in die Richtung „sepia" (Brauntöne) abwandelt. Des Weiteren wurde auf dieser Basis der Kontrast, die Helligkeit und einige Eigenschaften mehr variiert.
Das Gruppenbild hingegen wurde komplett individuell in den Eigenschaften Kontrast, Helligkeit und Wärme bearbeitet, bis es ein wenig comichaft anmutet.

Abbildung 58: bearbeitete Porträts und Gruppenbild

Diese wunderhübschen Fotos konnten nun mit großer Freude in die Unterseite „Team" zu jeder zugehörigen Beschreibung eines Teammitglieds ergänzt und die Platzhalter herausgeworfen werden.

Das Gruppenbild wurde in die neugestaltete Startseite mit eingefügt und befindet sich nun unter den großen Lettern, die beschreiben, von wem das Informatikprojekt bestritten wird.
Ebenso wurde ein Bild mit KillerRaupi vor Spaceballs-würdigem Hintergrund ergänzt, das nun den Blickfang unserer Startseite bildet.

Abbildung 59: Neues Design der Startseite

Weiterhin war kritisiert worden, dass die Animation auf der Website zu kurz kommt und deswegen wurde nun eine zusammengefasste Dokumentation über die Erstellung der Videos und Motive mit als neue Unterseite in die Website aufgenommen. Das Erscheinungsbild ist in der folgenden Abbildung zu sehen:

Abbildung 60: Unterseite mit Entwicklung der Animation

6.12 13. Projektwoche

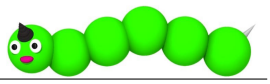
In dieser Woche ist es uns endlich gelungen ein erstes Video erst auf Youtube hochzuladen und anschließend mit unserer Website zu verlinken, was auch problemlos funktioniert hat. Nach dem Turnier werden hier natürlich die entsprechenden Turniervideos erscheinen, vielleicht nicht alle, aber auf jeden Fall eine Auswahl davon. Diese Seite wurde daraufhin auch noch ein wenig umdesignt, damit die Fenster der einzelnen Videos noch ein wenig größer erscheinen. Das Auge isst ja schließlich mit...

Abbildung 61: Verlinkungen zu den externen Videos

Weiterhin gab es noch eine Neuerung in der Anordnung der Unterseiten in der Menu-Leiste. Hier ist nun auch eine direkte Unterseite eigens für die Dokumentation zu finden. Zuvor war das nur über vereinzelte Links im Textverlauf von verschiedenen Unterseiten möglich, nun ist es auch möglich das komplette pdf-file herunterzuladen.

Abbildung 62: Menüleiste mit neuer Unterseite

In dieser Woche konnte nun endlich auch noch die Entwicklungsseiten vervollständigt werden, da sich ab jetzt in der Programmierung und der Website kaum noch etwas ändern wird.
Weiterhin wurde auch begonnen die Endversion des m-files und damit unserer KI in Worte zu formulieren. Das wird in der kommenden Woche noch vollendet werden.



6.13 Zusammenfassung

Die erforderliche Website für das Informatikprojekt wurde mit wix.com erstellt. Hierbei handelt es sich um einen Anbieter um Websites relativ unkompliziert und ohne Programmiersprache, sondern mithilfe einer visuellen Benutzeroberfläche zu designen. Per drag & drop lassen sich sehr ansprechende Websites erstellen, ohne die Computersprache HTML5 beherrschen zu müssen.
Unsere Website kann über den folgenden Link erreicht werden: http://killerraupi.wix.com/killerraupi

Unsere Website besitzt in ihrer Endversion nun 13 Seiten mit einigen Texten aber vor allem auch sehr vielen Bildern. Auf der Startseite begrüßt den Besucher eine freundliche Überschrift zusammen mit einem Bild von KillerRaupi in spaceballhafter Umgebung. Weiterhin ist unser Gruppenbild zu sehen, sowie die Klassifizierung als Website eines Informatikprojektes. Als besonderes Feature schweben beim Erscheinen der Startwebsite das Logo und der Text „KillerRaupi" in die linke obere Ecke ein.

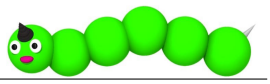
Der zweite Reiter bietet Unterseiten zum Projekt im allgemeinen und mit unserer Umsetzung. Die Unterseite „Spaceballs" beschreibt den Ihnhalt des Projektes und geht auf unsere grundsätzliche Idee der Umsetzung ein.
Auf der folgenden Unterseite „Spiele" sind die aufgezeichnete Spiele unserer KI aus dem finalen Turnier zu sehen. Ebenso haben wir es mit der nächsten Seite der Website „Animation" gehalten, auch dort finden sich Links, allerdings diesmal zu den Videos, die für Spaceballs speziell mit KillerRaupi animiert wurden.
Auch auf der nächsten Unterseite dreht es sich noch um Animation, dort findet man von der 1. Projektwoche an die Fortschritte der Animation kurz beschrieben.

Die Unterseiten des dritten Reiters beschäftigen sich mit unserem Projektteam. Auf der ersten Unterseite „Team" findet sich noch einmal unser Gruppenbild und dann zu jedem Mitglied ein Porträtfoto nebst einem kleinen Text der die Aufgabe im Projekt und die persönlichen Fortschritte beinhaltet.
Die nächste Unterseite „spezielle Teammitglieder" ist witzigen Raupengästen aus der Projektzeit gewidmet.

Der vierte Reiter bietet Einblicke in unsere Figur „KillerRaupi". Die erste Unterseite hierbei beinhaltet eine Bildergalerie mit verschiedenen Motiven unserer Raupe. Hinter der nächsten Unterseite mit dem Namen „Tagebuch" verbirgt sich die Entstehung der KI über die gesamten Projektwochen aus Sicht unserer KillerRaupi.
Eine Zusammenfassung dieses Prozesses und die finale Fassung der KI sind in der letzten Unterseite dieses Reiters „Endversion der KI" zu sehen und zu lesen.

Der fünfte Reiter mit Titel „Dokumentation" beinhaltet einen Link zu unserer Dokumentation als pdf-file, in der der gesamte Hergang und Entwicklungsfortschritt aller Projektwochen nachzulesen ist.

Der sechste und letzte Reiter bietet die Unterseite „Kontakt". Da der Name ja Programm ist, findet man auf dieser Seite ein Kontaktformular wo man der Gruppe Kil-



lerRaupi mit Name, Email, Betreff und einer entsprechenden Nachricht mit Anregungen und/oder Fragen übermitteln kann.
Schlussendlich und wie es sich gehört findet sich auf der letzten Unterseite noch ein Impressum, wo noch einmal alle Teammitglieder aufgelistet sind, sowie eine Kontaktadresse zu finden ist und die Haftungsbestimmungen aufgeführt werden.

7 Turnier und Fazit

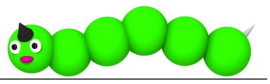
7.1 Soll-Ist-Vergleich

Wir haben nicht alles erreicht was wir uns vorgenommen haben.
Wahrscheinlich ist es strategisch nicht gut mit diesem Satz einzuleiten, aber so ist es nun einmal einfach. Wir haben es leider nicht geschafft unseren Spaceball mit einem großen Schach-Kalkül auszustatten und exakt den perfekten Weg für jedes Spielfeld berechnen zu lassen. Ebenso ist es uns nicht direkt gelungen mehrere Gefahren für unseren Spaceball zu verketten und zusammenzufügen. Wir haben auch nicht eine hundertprozentige Variante Minen zuverlässig zu umfahren gefunden, insbesondere wenn die Minen alle hintereinander und kombiniert auftauchen. Das steht leider in sehr starkem Kontrast zu unserer anfänglichen Zielsetzung nicht durch vermeidbare Fehler sich selbst abzuschießen.
Wir können tanken. Wir können sogar ausgezeichnet tanken und sind in der Lage, sofern es das Spielfeld hergibt zwei Minen auf einen Schlag zu leeren. Diese Zielstellung ganz zu Beginn des Projektes haben wir nach Meinung dieser Dokumentarin vollständig und Abstriche zu machen erfüllt und uns damit eine sehr gute Ausgangsposition im Turnier erschaffen. Wir können zuverlässig angreifen, ohne dabei jedoch dem Angriff zu viel Bedeutung beizumessen, ebenfalls wie wir uns vorgenommen haben. Unsere Strategie stellt sich tatsächlich am Ende eher recht passiv dar, wir verschwenden keinen Tropfen Sprit - wenn es nicht volle Absicht ist - lauern dem Gegner lieber vor der letzten Tankstelle auf als offen in den Krieg zu ziehen und bleiben eben auch einfach stehen, wenn wir größer sind als der Gegner je werden kann. Wir haben nur eine bescheidene Verteidigung, die wir aber vermutlich nicht brauchen werden, weil wir in der Lage sind fehlerfrei zu tanken.

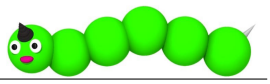
Wir haben uns dann auch noch entschieden das Gewinnen-Video nicht zu splitten in eine Sprit- und Angriffs-Variante. Unsere Raupe frisst keine Blätter, sie wird von einer Trinkflasche befüllt. Aber, wir haben insgesamt sechs Clips animiert, mit einer durchweg niedlichen Raupe in der Hauptrolle, die entgegen unserer Vorsätze, wirklich einfach nur süß ist. Das einzige, das ein wenig bedrohlich sein könnte, wäre wohl das gezwirbelte Einhorn-Horn. Alle Videos sind mit Ton ausgestattet. Darüber hinaus haben wir ein tolles Instrumentalstück als Intro, selbst aufgenommen. KillerRaupi hat sogar ein wunderhübsches kleines Haus bekommen. Sie hat Emotionen von der Animation erhalten, kann Weinen und Schluchzen. Aber das Wichtigste ist: Sie kann sich in einen Schmetterling umwandeln.

Unsere Website ist genau so geworden wie wir sie uns vorgestellt haben, nur noch ein bisschen schöner und mit noch viel mehr Inhalten. Das beginnt mit verlinkten Videos, geht über eine Kurzfassung der Dokumentation bis hin zu einer umfangreichen Vorstellung unserer erdachten Figur KillerRaupi. Wir haben eine ganz Welt um unsere Figur erdacht und mehr als einmal haben wir uns im Alltag gewundert wo sich überall Raupen finden.

Entgegen unserer anfänglichen Erwartungen ist es nicht vorrangig wichtig das Turnier für uns zu entscheiden. Viel wertvoller war es eine derartig umfangreiche, ver-



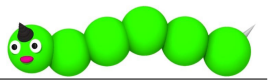
netzte Arbeit ausführlich und intensiv zu erschaffen und dabei als Team zusammen zu wachsen. Wir sind stolz auf uns und auf das was wir geschafft haben, egal ob unser Gruppenname auf der Ergebnisliste ganz oben stehen wird, einfach weil wir es alle zusammen mit viel Liebe zum Detail und Ehrgeiz über einen langen Zeitraum geschafft haben das Projekt mit Leben zu erfüllen.



7.2 Turnier, Erfolg unserer KI un Fehlerauswertung

Das Turnier war rege besucht, fast alle Mitglieder der teilnehmenden Gruppen waren da und sogar einige Studierende aus anderen Bereichen und dem nachfolgenden ILST wollten sich diesen Wettkampf nicht entgehen lassen. Nach begrüßenden und einleitenden Worten des betreuenden Professors ging es auch direkt mit der Vorstellung aller Gruppen los. Die Porträts der Teams untermalt mit dem jeweiligen Intro werden eingeblendet und anschließend alle erstellten Clips gezeigt. Das ist für das Publikum ausgesprochen unterhaltsam, die Stimmung ist sehr gut und alle sind gespannt auf die Spiele. 72 an der Zahl laufen in den nächsten zwei Stunden als Aufzeichnungen ab (Live-Simulation kam nicht in Frage, da dadurch das Zeitlimit überschritten worden wäre). Nach dieser Zeit steht der Gewinner mit einer Höchstpunktzahl von 21 Punkten fest. Herzlichen Glückwunsch, wir sind es nur leider nicht. Unsere liebe KillerRaupi hat es nur auf Platz acht geschafft, den vorletzten Rang, mit 4 gewonnenen Spielen.
Der Erfolg unserer KI hält sich leider in Grenzen und es ist nicht schwer unseren größten Fehler zu entdecken: Wir haben uns einfach zu sehr darauf verlassen, dass wir besser tanken als der Gegner und entsprechend planlos ist unser Spaceball immer dann herum gefahren, wenn er eben nicht mehr Sprit hatte als der Gegner, was leider sehr oft vorkam. Zweimal sind wir zwischen sehr großen Minen hängen geblieben, das war einem Sieg auch nicht gerade förderlich. Ein ganzes Stück Pech war leider auch dabei, denn häufig lagen die Tankstellen weit auseinander und hinter großen Minen versteckt, sodass wir viel Sprit durch umfahren vergeudet haben. Hinzu kam noch, dass wir mehr als einmal dem Gegner geradewegs in die Arme gefahren sind, da wir nicht definiert hatten, was unser Spaceball tun soll, wenn er weniger Sprit hat als der Gegner und alle Tankstellen aufgebraucht sind. Dann sind wir nur noch durch die Gegend geglitten, meistens leider geradewegs in den Gegner hinein, dass wir dann keine umfangreiche Verteidigung hatten hat auch nicht gerade geholfen.
Ein ganz bitterer Verlust war jedoch ein Spiel, indem wir gut doppelt so groß wie der Gegner waren und auf einmal in eine Mine hineingefahren sind. Erklären konnten wir es uns alle nicht, seit es das Minensystem gab sind wir in keine einzige Mine mehr hineingefahren und der Angriff wurde auch nur dann eingeleitet, wenn der Weg zum Gegner frei von Minen war.
Wahrscheinlich hätten sich einige dieser Fehler vermeiden lassen, wenn wir beim Ausprobieren unserer KI immer nur gegen die vorgefertigte KI unseres Professors angetreten wären, sondern gegen uns selbst. Dann hätten wir schnell gemerkt, dass wir noch einige Befehle mit einbauen sollten, die unserem Spaceball sagen wie er sich verhalten soll wenn der Gegner ähnlich viel Treibstoff hat wie wir.
Alle Spiele unserer KillerRaupi können auf unserer Homepage angeschaut werden:
http://killerraupi.wix.com/killerraupi#!spiele/c1w25

Auch wenn es für uns als Gruppe persönlich nicht so gut ausgegangen ist wie wir es uns erhofft hatten, so war das Turnier auf jeden Fall sehr spannend und unvergesslich.



Abbildungsverzeichnis