Maschinelles Spielen im Klassiker Asteroids

Paul Czerwionka, Marco Block, Maro Bader und Raúl Rojas

Freie Universität Berlin
Institut für Mathematik und Informatik
Takustr. 9, 14195 Berlin, Germany
{czerwionka,block,bader,rojas}@inf.fu-berlin.de

Zusammenfassung Das Ende der 80er Jahre veröffentlichte, auf Reaktion und Planung basierende Spiel *Asteroids*, gilt inzwischen als eines der erfolgreichsten Computerspiele aller Zeiten. Es eignet sich hervorragend als Testplattform für die verschiedensten KI-Konzepte, die anhand erreichter Punktezahlen verglichen werden können. In der vorliegenden Arbeit wird ein adaptives physikalisches Weltmodell eingeführt, mit dessen Hilfe eine optimale Verhaltensstrategie gefunden werden kann. Mittels einer heuristischen Suche wird eine Aktionsfolge gefunden, die in minimaler Zeit versucht, die Punkte zu maximieren. Die hier vorgestellten Ideen wurden im Programm *jAsteroidsAI* realisiert, welches erfolgreich am Wettbewerb der Computerzeitschrift c't 2008 teilnahm und in fünf Minuten eine Spitzenpunktezahl von 85.000 erreichte.

1 Einführung und Motivation

Das Spiel *Asteroids* wurde 1979 von Lyle Rains und Ed Logg für Atari in einem Arcade-Spielautomat umgesetzt. Mit über 70.000 verkauften Automaten und Portierungen auf alle gängigen Computersysteme zählt es zu den größten Erfolgen aller Zeiten in der Geschichte der Computerspiele (siehe dazu Abbildung 1) [5].

Das ursprünglich für menschliche Spieler enwickelte Reaktionsspiel bietet mit einer diskretisierbaren Welt und den nur 5 möglichen Aktionen (Links-/Rechtsdreh, Schuss, Impuls, Raumsprung) eine gute Gelegenheit, Konzepte für einen künstlichen Spieler zu testen. Aufgrund der Zufallselemente im Spiel ist es auch bei unbegrenzter Suchzeit nicht möglich, eine eindeutig beste Strategie zu finden. Dies begünstigt den Einsatz heuristischer Entscheidungsfunktionen bei der Suche nach einer optimalen Aktionsfolge.

2 Das Spiel *Asteroids*

Ein vom Spieler kontrolliertes Raumschiff befindet sich dabei in einer rechteckigen Weltraumszene mit dem Ziel, die sich auf zufälligen Bahnen im Raum mit unterschiedlicher Geschwindigkeit bewegenden Asteroiden und gegenerischen Ufos abzuschießen und dabei Kollisionen mit diesen zu vermeiden. Bestimmte Asteroiden (mittlere und große) zerfallen nach einem Abschuss in zwei kleinere

Abbildung 1. Original-Konsole des Spiels *Asteroids* (Abbildung aus [6])

Einheiten, welche auch zufällige Flugbahnen beginnend beim Abschussort einnehmen. Für jeden Abschuss erhält der Spieler Punkte (siehe Abbildung 2).

Der sichtbare Weltraumbereich von *Asteroids* hat die Eigenschaften eines Torus: Objekte, die über eine der vier Kanten die Szene verlassen, erscheinen mit gleicher Ausrichtung und Flugbahnvektor an der gegenüberliegenden Kante. Dies gilt auch für die Schüsse, von denen zu keinem Zeitpunkt mehr als 4 auf dem Bildschirm sichtbar sein können.

Während des Zerfalls der Asteroiden nach Abschüssen wird von dem Programm reguliert, daß sich zu keinem Zeitpunkt mehr als 27 Asteroiden in der Weltraumszene befinden.

In unregelmäßigen Zeitabständen erscheint ein großes oder kleines UFO, welches einmalig horizonal durch den Weltraumbereich fliegt ohne an der gegenüber liegenden Kante wieder zu erscheinen. Ufos feuern Schüsse ab, denen das Schiff durch Beschleunigen oder Raumsprünge (*hyperspace*) ausweichen kann.

3 Der *Asteroids* Wettbewerb der Computerzeitschrift *c't*

Von der Computerzeitschrift *c't* [8] wurde zu Beginn des Jahres 2008 ein Wettbewerb ausgerufen, bei dem es galt, einen künstlichen Spieler für das Spiel *Asteroids* einzureichen. Zum Sieger wurde in einem mehrstufigen Verfahren das Programm auserkoren, welches innerhalb von 5 Minuten die meisten Punkte erreichen konnte. Ausgeführt wurde der Wettbewerb innerhalb des Emulator MAME[9].

Es handelt sich um ein open-source Emulator, für welchen die ROM Programmdatei von *Asteroids* von den Ausrichtern des Wettbewerbs bereitgestellt

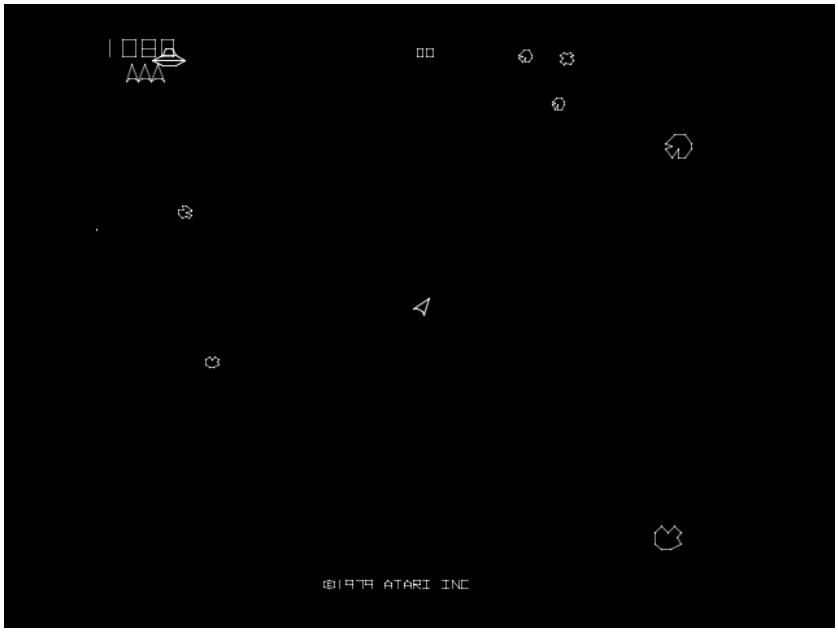

Abbildung 2. Screen des Spiels (Abbildung aus [6])

wurde. Für die Anbindung an andere Computersysteme wurde MAME so angepasst, daß es 60 mal pro Sekunde den aktuellen Inhalt des RAMs per universal datagramm package (UDP) ins Netzwerk sendet. Die Spielereingaben werden ebenfalls per UDP an den Emulator übermittelt.

4 Das Programm *jAsteroidsAI*

Das Programm *jAsteroidsAI* stellt einen künstlichen Spieler für *Asteroids* bereit. Es liefert eine auf einem physikalischen, adaptiven Modell der Weltraumszene basierende Strategie, die versucht, in möglichst minimaler Zeit eine maximale Punkteausbeute zu erreichen.

Als Rahmen wurde die allgemeine Spielebibliothek *jGameAI* verwendet, welche neben generischen Suchfunktionen eine graphische Darstellung des Weltbildes und die Untersuchung der KI-Konzepte ermöglicht [2].

Beide Projekte werden fortlaufend in der Arbeitsgruppe Spieleprogrammierung der Freien Universität Berlin weiterentwickelt.

4.1 Physikalisches Modell

Alle Objekte beschreiben einen Impuls, der nicht aus dem Datenpaket ausgelesen werden kann und somit manuell bestimmt werden muss. Das Schiff besitzt

zusätzlich eine Aus- und eine Schussrichtung, die leicht von einander abweichen können.

Für das Weltbild ist es notwendig, Objekte in zwei aufeinanderfolgenden Datenpaketen wieder zu identifizieren. Daraus lässt sich der Impuls eines sich bewegenden Objekts und die möglichen Abschüsse und Entstehungen neuer Objekte ermitteln. Mögliche Paketverluste und Latenzprobleme müssen bei der Weltbildberechnung und der Suche nach einer optimalen Strategie berücksichtigt werden.

Modell für Positions- und Verhaltensvorhersagen

Zuerst gilt es in zwei aufeinander folgenden Zeitschritten alle dargestellten Objekte wieder zu erkennen. Naheliegend geschieht dies über die Raumkoordinaten. Die zusätzlichen Eigenschaften, Größe und Typ erlauben eine eindeutige Wiedererkennung. Nicht zugeordnete Objekte müssen demnach neu erschienen sein. Alle Objekte eines vorhergehenden Zeitschrittes, die nicht neu zugewiesen werden konnten, werden aus dem internen Weltbild entfernt.

Alle Objekte behalten ihre ursprünglich zugewiesene Geschwindigkeit für die gesamte Lebensdauer bei. Zusammen mit der Bewegungsrichtung können somit Vorhersagen zukünftiger Positionen getroffen werden. Dies wird als Impulsvektor für alle sich bewegenden Objekte festgehalten. Zusätzlich dazu werden eine feste Anzahl N an zukünftigen Positionen mitgeführt.

Dieser lässt sich anhand der Positionsveränderungen mittels linearer Regression über die zwei Raumkoordinaten berechnen. Es ist dadurch möglich, zukünftige Positionen bei gradliniger Bewegung zu ermitteln. Idealerweise entspricht der nächste Vorhersagepunkt der wiedergefundenen Position eines Objektes. Ist dies nicht der Fall, muss eine Korrektur des Impulsvektors durchgeführt werden.

Eine Reinitialisierung des Impulsvektors muss bei den möglichen Flugbahnänderungen der Ufos durchgeführt werden. Ufos ändern ihre Bewegungsrichtung in bekannten Winkeln, so dass eine solche leicht erkannt werden kann.

Um Artefakte bei Sprüngen über die Ränder zu umgehen, wird für jedes Objekt ein auf dieses zentrierte Koorditensystem verwendet. Für die vorhergesagten Positionen findet eine Rücktransformation in die rhombischen Weltraumkoordinaten statt, um somit Kollisionen mit anderen Objekten identifizieren zu können.

Um beispielsweise den Abschuss eines Asteroidens zu berechnen, genügt es alle sein Vorhersagepunkte mit denen eines Schusses aus der aktuellen Schiffsposition miteinander zu vergleichen. Die erste Übereinstimmung innerhalb eines Abstands kleiner dem Radius des Asteroiden entspricht dem Zeitpunkt des Abschusses.

Raumschiffausrichtung und Schussrichtung

Innerhalb eines erhaltenen Datenpaketes wird die neben der Position des Schiffes auch seine Ausrichtung mitgeführt. Diese entspricht der graphischen Darstellung und kann leicht von der der manuell zu bestimmenden Schussrichtung abweichen. Untersuchungen ergaben, dass es 256 unterschiedliche Schussrichtungen gibt, die in einer Tabelle festgehalten werden.

Nach einem Rotationsschritt des Schiffes verändert sich die Schussrichtung innerhalb der Tabelle um eine Zeile. Es liegen nur 128 unterschiedliche Schiffssausrichtungen vor, so dass eine Änderung dieser erst nach jeweils zwei in die gleiche Richtung unternommenen Rotationsschritten stattfindet.

Ingesamt verteilen sich die Schussrichtungen auf drei vollständige Umdrehungen des Schiffes. Es ist daher möglich, dass um einen bestimmten Punkt im Raum zu treffen, bis zu $1\frac{1}{2}$ Umdrehungen durchgeführt werden müssen.

4.2 Optimale Schussreihenfolge in *jAsteroidsAI*

Da zu jedem Zeitpunkt des Spiels nur begrenzte Handlungsmöglichkeiten zur Verfügung stehen, handelt es sich bei den scheinbar kontinuierlichen Suchraum, um einen diskreten. In diesem lassen sich mit bewährten Suchmethoden nach der optimalen Reihenfolge der Aktionen, eine Strategie, suchen. Jeder Zeitpunkt entspricht dabei einem Spielframe der Simulation. Im Wettbewerb besteht daher ein vollständig durchgespielter Durchgang aus 18.000 Frames (300 Sekunden à 6 Frames)

Als Spielerzug l wird im folgenden die Aktionsfolge $l = a_1, a_2 \ldots, a_n$ bezeichnet, die n Zeitschritte in Anspruch nimmt und den nach einander ausgeführten Handlungen a_i entspricht. Eine Spielerzug beinhaltet beispielsweise der benötigte Rotation, eingeschobene Warteframes und dem anschließenden Abschuss eines Asteroiden oder Ufos. Der Spielerzug l^i_{opt} beschreibt die kürzeste Aktionsfolge, die für den Abschuss des i. Weltraumobjekts benötigt wird..

Eine Strategie besteht aus einer Reihenfolge von kürzesten Aktionsfolgen. Die durch die Strategie erreichte Punktezahl entspricht dabei den gewonnen Punkten aller Spielerzüge.

Das Finden der optimalen Strategie in *Asteroids* ist als ein Entscheidungsproblem bezüglich der Zeit und den erreichten Punkten formalisiert worden, das mit Hilfe des Suchbaumbasierten Verfahrens von jGameAI gelöst werden kann. Es handelt sich dabei um eine leicht zu integrierende KI-Funktionsbibliothek, die von der Arbeitsgruppe Spieleprogrammierung entwickelt worden ist.

Die Minimale Aktionsfolge l_{opt}

Für jeden Asteroiden A_i kann eine minimale Aktionsfolge l^i_{opt} bestimmt werden. Diese setzt sich zusammen aus: Drehung, Warteframes und Schuss. Für alle N zukünftigen Zeitschritte t^i von A_i, werden die benötigten Rotationsschritte s^i_{rot} und die Schussdauer s^i_{shot} bis zum Abschuss berechnet.

Der Wert s^i_{rot} wird dabei einfach aus der Tabelle abgelesen: er entspricht der Differenz zwischen aktueller Schussrichtung des Schiffes und der benötigten Schussrichtung zu A_i. In den häufigsten Fällen müssen aufgrund der diskreten Schussrichtungen des Schiffes nach einer Rotation noch s^i_{wait} Warteframes eingebaut werden. Diese lassen sich über die Formel $s^i_{wait} = t^i - (s^i_{rot} + s^i_{shoot})$ berechnen. Ist s^i_{wait} negativ, kann A_i nicht zum Zeitpunkt t abgeschossen werden.

Minimal ist die Aktionsfolge zum Zeitpunkt t^i_{hit}, für die s^i_{wait} zum ersten Mal ≥ 0 ist. Die Länge t^i_{opt} von l^i_{opt} ist kleiner als t^i_{hit}, da sie nicht die Schussdauer beinhaltet. Es ist wichtig zu verstehen, dass bereits nach t^i_{opt} Zeitschritten nach weiteren Aktionsfolgen gesucht werden kann, da nicht bis zum Abschuss des Asteroiden gewartet werden muss, bis neue Handlungen durchgeführt werden können.

Die Latenz beschreibt die zeitliche Verzögerung zwischen dem Senden von Aktionen und dem Empfang der Ausführungsbestätigung durch MAME. Diese beträgt mindenstens zwei Zeitschritte und kann bei Überstragsproblemen höhere Werte annehmen. Bei der Berechnung des frühstmöglichen Abschuss eines Objektes, können nur Zeitpunkte in Betracht gezogen werden, die ferner als die aktuelle Latenz in der Zukunft liegen ($t > Latenz$).

Lösung des Entscheidungsproblem mit Bestensuche

Mit der vereinfachenden Annahme, dass nach dem Abschuss eines Asteroiden keine neuen entstehen, entspricht die beste Strategie der Reihenfolge minimaler Aktionsfolgen l^i_{opt}, die sich am schnellesten ausführend lässt, so dass alle zu Beginn der Suche sichtbaren Weltraumobjekte A_i abgeschossen werden.

Zwischen Aktionsfolgen müssen eventuell noch zusätzliche Warteframes eingebaut werden. Da zu keinem Zeitpunkt mehr als vier Schüsse sichtbar sein können, muss bis zum Einschlag des ersten abgefeuerten Schusses gewartet werden.

Die nach einem Abschuss neu erscheinenden Asteroiden führen allerdings in den häufigsten Fällen dazu, dass die bisher beste Strategie obsolet wird. Diese befinden sich in der Nähe des abgeschossenen Asteroiden und müssen in einem Aktualisierungsschritt der Abschussreihenfolge oft weit nach vorne plaziert werden. Eine Planung über den ersten Abschuss hinweg ist daher wenig sinnvoll. Ähnliches geschieht bei Auftreten eines Ufos, der aufgrund seiner hohen Punktezahl auf jeden Fall berücksichtigt werden muss.

Daher wird in *jAsteroidsAI* nur jeweils die nächste optimale Aktionsfolge l^i_{opt} als gefundener Spielerzug zurückgeliefert. Findet die KI keine Aktionsfolge bis zum nächsten Takt, so wird die Suche abgebrochen, das Weltbild mit dem folgenden Datenpaket aktualisiert und eine neue Suche gestartet. Ansonsten liegt l^i_{opt} vor und die nächste Handlung wird übermittelt. Nach Durchlaufen aller Handlungen von l^i_{opt} wird im drauf folgenden Zeitschritt eine neue Suche gestartet.

Neben dem suchenden Prozess führt *jAsteroidsAI* ebenfalls einen Datenaustauschprozess durch. Dieser garantiert eine korrekte Einhaltung des von MAME verwendeten UDP-Protokolls, welches sechs Mal in der Sekunde ein Datenpaket mit den Koordinaten aller sichtbaren Raumobjekte sendet und anschliessend für dem Empfang der Spielereingabe bereit steht ..

Weitere Heuristiken: Dreifachschuss, Raumsprung und Punktepriorität

Für den Abschuss großer und mittlerer Asteroiden bietet sich an, mehrere Schüsse in Form von Salven abzufeuern, mit der Absicht die neu erscheinenden Asteroiden ebenfalls zu erwischen. Es hat sich gezeigt, dass das Feuern von drei Schüssen dafür ausreicht.

Da Ufos wesentlich mehr Punkte als Asteroiden einbringen erhalten, sie bei der Suche nach der nächsten Aktionsfolge immer den Vorzug. Dies gilt auch für Asteroiden, die zwar in gleicher Zeit abgeschossen werden könne, aber unterschiedliche Punkte einbringen.

Um den Verlust des eigenen Schiffes zu vermeinden, muss vor jeder neuen Suche ermittelt werden, ob eine eigene Kollision mit einem Asteroiden, einem Ufo oder dessen Schuss bevorsteht. Tritt diese früher ein als t^i_{opt} besteht die letzte Rettung in dem Ausführen eines Raumsprungs. Anschliessend muss bei Wiederauftauchen des Raumschiffes eine neue Suche durchgeführt werden.

5 Resultate und Ausblick

Das Programm *jAsteroidsAI* erreichte in den besten Durchgängen der 5-Minuten Spielvariante bis zu 85.000 Punkte und gemittelt auf 200 Testläufen eine durchschnittlich knapp 79.000. Das verwendete physikalische Modell wurde mit Hilfe der *jGameAI* graphisch aufbereitet und untersucht (siehe auch Abbildung 3).

Das Projekt wird zu Testzwecken weiterer Konzepte und wissenschaftlichen Arbeiten an der Freien Universität zu Berlin von der Spieleprogrammierung-Arbeitsgruppe weitergeführt. Es bietet sich für praktische Arbeiten innerhalb akademischer Veranstaltungen an. Die Verwendung im Rahmen der Lehrveranstaltung "Künstliche Intelligenz" an der Freien Universität Berlin ist bereits geplant.

5.1 Wieso nicht gewonnen?

Gegenüber den Siegern des Wettbewerbes, die bis zu 145.000 Punkte erreichten erscheint das hier vorgestellte Konzept schlecht. Es gilt allerdings zu beachten, dass fast alle Teilnehmer den Zufallsgenerator des *Asteroids*-ROM-Programm geknackt haben. Das zufällige Erscheinen von Asteroiden, Ufos und deren Impulse konnte deterministisch vorhergesagt werden.

Es ist mehr als fraglich, ob ein Vergleich zu diesen Lösungsansätzen Sinn ergibt, da die wesentliche Eigenschaft des Zufallselementes von *Asteroids* ausfällt. Ein neuer Wettbewerb, der dies verhindert, könnte aussagekräftigerere Aufschlüsse über die eigene Spielstärke liefern.

5.2 Erweiterungen und Verbesserungen

Viele Erweiterungen bieten sich zur Verbesserung der Spielstärke an. So berücksichtigt die hier vorgestellte Lösung keine Aktionsfolgen, die eine Eigenbewegung

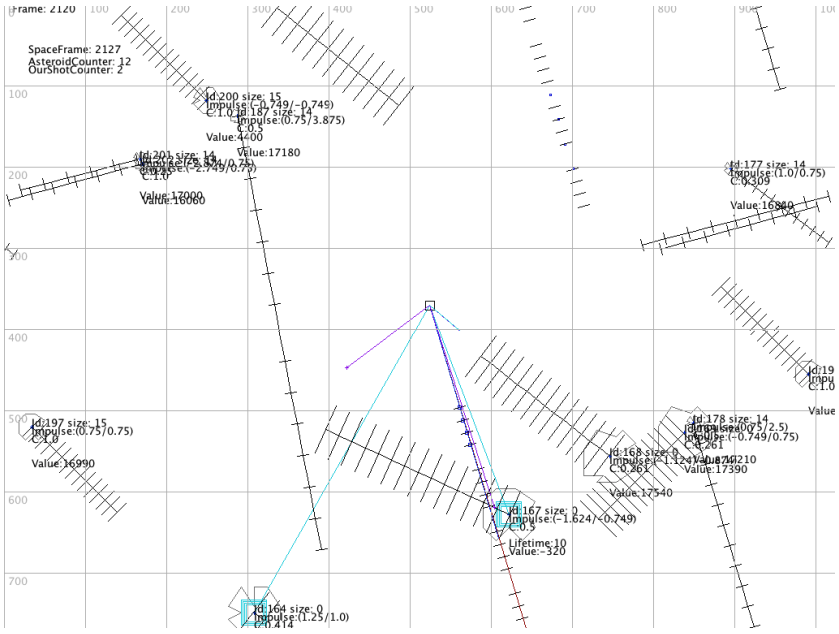

Abbildung 3. Eine typische Spielszene mit allen von *jAsteroidsAI* verwerteten zusätzlichen Informationen: Impuls, Objektyp, Identifizierung und Position.

des Raumschiffes beinhalten. Dafür müsste eine dynamische Kollisionserkennung stattfinden und ein favorisierter Raumbereich berechnet werden, dessen Anflug sich durch schnellere Abschüsse lohnt. Vereinzelt konnten bei einigen Teinehmern des Wettbewerbs festgestellt werden, dass sie sich in die Ecken, der Weltraumszene positionierten, da hier der mittlere Abstand zu allen Weltraumobjekten am kleinsten ist, was auf dir Torus-Eigenschaften des Raumes zurückzuführen ist.

Da es in vielen Applikationen schwierig ist, basierend auf einer einzigen Entscheidungsfunktion die korrekte Wahl der nächsten Aktionsfolge zu bestimmen, ist eine automatische Stellungsklassifizierung geplant. Diese besitzt für unterschiedliche Spielsituationen in Abhängigkeit von Gefahren und Anzahl sichtbarer Objekte eine parametrisierbare Entscheidungsfunktion. Diese Parameter können durch den Einsatz von Temporaler Differenz kalibiriert werden. Diese beiden Techniken wurden bereits erfolgreich bei dem Schachprogramm FUSc# angewendet [1].

Literatur

1. Block M., Bader M., Tapia E., Ramírez M., Gunnarsson K., Cuevas E., Zaldivar D., Rojas R.: "*Using Reinforcement Learning in Chess Engines*", CONCIBE SCIENCE 2008, Guadalajara/Mexico, 2008

2. Bader M.: "*Eine allgemeine selbstlernende Strategie für nicht-kooperative Spiele*", Diplomarbeit (in Bearbeitung), Freie Universität Berlin, 2008.
3. Bögeholz H.: "*Programmierwettbewerb zum 25. c't-Geburtstag*", Programmierwettbewerb creativ'08, Computerzeitschrift c't Vol. 9/2008, p.176, 2008
4. Wüthrich H.: "*Emulatoren: Wie Computersysteme und Spielkonsolen unsterblich werden*", Skriptorium-Verlag, ISBN: 978-3938199084, 2. Auflage, 2007
5. Asteroids-Beschreibung bei Wikipedia: *http://de.wikipedia.org/wiki/Asteroids*
6. Klov-Webseite: *http://www.klov.com/game_ detail.php?letter=&game_ id=6939*
7. Webseite von Heise: *http://www.heise-medien.de/default.php/ mediengruppe,zeitschriften/11/2*
8. Wettbewerb bei der c't: *http://www.heise.de/ct/creativ/08/02/*
9. MAME-Webseite: *http://de.wikipedia.org/wiki/MAME*

Sofern nicht anders angegeben, waren alle URLs gültig am 27.Oktober 2008.