

INHALTSVERZEICHNIS

VORWORT

1 ZELLULÄRE AUTOMATEN

- 1.1 Erklärung und Definition
- 1.2 Merkmale
 - 1.2.1 Geometrie der Zellanordnung
 - 1.2.2 Nachbarschafts-Konstellation
 - 1.2.3 Anzahl der möglichen Zellzustände
 - 1.2.4 Zellregeln

2 LIFE - DAS SPIEL DES LEBENS

- 2.1 John Horton Conways „Game of Life“
- 2.2 Das erste selbstreproduzierende Muster von Edward Fredkin
- 2.3 Carter Bays' Life 4555 und Life 5766

3 PROGRAMMBESCHREIBUNG

- 3.1 Hauptbildschirm
 - 3.1.1 Hauptfenster
 - 3.1.1.1 Statusfenster
 - 3.1.1.2 Positionsschaltflächen
 - 3.1.1.3 Zoomschaltflächen
 - 3.1.1.4 Distanzschaltflächen
 - 3.1.1.5 Funktionsschaltflächen
 - 3.1.2 Menüleiste
 - 3.1.2.1 Datei
 - 3.1.2.1.1 Menübefehl „Neu“
 - 3.1.2.1.2 Menübefehle für Ein- und Ausgabe von Dateien
 - 3.1.2.1.3 Menübefehl „Programm beenden...“
 - 3.1.2.2 Optionen
 - 3.1.2.2.1 Menübefehl „3D-Raster“
 - 3.1.2.2.2 Menübefehl „Zellrahmen“
 - 3.1.2.2.3 Menübefehl „Zelluniversum“
 - 3.1.2.2.4 Menübefehl „Zellabstand einstellen...“

- 3.1.2.2.5 Menübefehl „Nachbarn auswählen...“
 - 3.1.2.2.6 Menübefehl „Datum auf Null setzen“
 - 3.1.2.3 Editor
 - 3.1.2.4 Konfiguration
 - 3.1.2.5 Info
 - 3.2 Zelleditor
 - 3.2.1 Titelleiste
 - 3.2.2 Funktionsschaltflächen
 - 3.2.3 Funktionsfenster
 - 3.2.4 Bereich für die zweidimensionale Darstellung
 - 3.2.5 Bereich für die dreidimensionale Darstellung
 - 3.3 Konfigurationseditor
 - 3.3.1 Titelleiste
 - 3.3.2 Zellschaltflächen
 - 3.3.3 Zellregelnfenster
 - 3.3.4 Funktionsschaltflächen
 - 3.3.5 Integrierte Funktionen
 - 3.3.6 Wertigkeitsfenster

ANHANG A QUELLTEXT PROGRAMM „LIFE 3D“

ANHANG B QUELLTEXT UNIT „MAUSMENU“

ANHANG C QUELLTEXT UNIT „FASTTEXT“

ANHANG D LITERATURVERZEICHNIS

ANHANG E ANDERE HILFSMITTEL

ANHANG F KALENDARIUM

VORWORT

Als ich zum ersten Mal den Begriff „zelluläre Automaten“ zu Ohren bekam, dachte ich zuerst an eine neue Errungenschaft in der Spielebranche, weil ich das Wort „Automat“ mit einem Spielautomat, der in fast jeder zweitklassigen Kneipe vorzufinden ist, in Verbindung brachte. Weit gefehlt, wie sich später herausstellen sollte. Zu diesem Zeitpunkt wußte ich den Begriff „zelluläre Automaten“ nirgends einzureihen. Doch gerade diese Unwissenheit machte mich neugierig und weckte in mir das dringende Bedürfnis, mich näher über dieses Thema zu informieren. Ich war sehr erstaunt, als ich erfuhr, was zelluläre Automaten eigentlich sind und gleichzeitig fasziniert über die Optik der komplexen Gebilde, die schon mit primitivsten zellulären Automaten hervorgebracht werden können.

Das eigentliche Interesse an diesem Thema erwuchs in mir aber erst durch die ungeheuer große Zahl der möglichen Regeln, die einen zellulären Automaten beschreiben können. Zum Beispiel ergeben sich für einen zweidimensionalen binären Automaten (die Zellen können nur zwei Zustände einnehmen) mit 8 Nachbarn bereits mehr als 10^{77} verschiedene Regeln. Nur ein winziger Bruchteil konnte bis heute untersucht werden. Mit meinem Programm „Life 3D“ möchte ich einen Teil zur Erforschung von zellulären Automaten beitragen. In „Life 3D“ können Regeln für zelluläre Automaten zusammengestellt werden, die nur durch die Phantasie des Benutzers begrenzt sind, und die Evolution der den Regeln unterworfenen Zellanordnungen, die mit dem im Programm integrierten Editor erstellt werden, kann man sofort beobachten und die oft komplexen, oft überraschend difusen, aber symmetrischen Zellgebilde bestaunen. Zelluläre Automaten, die zu Beginn nur durch ihre Unvorhersehbarkeit bewundert und zum Generieren von exotischen Mustern verwendet wurden, avancierten mit der Zeit zu Modellen der realen Welt, sei es zur Beschreibung des Wachstums von einfachen oder dendritischen Kristallen, sei es zur Simulation von bestimmten Diffusionsreaktionen. Der Anwendungsbereich von zellulären Automaten ist seit ihrer Erfindung, die zur Zeit der ersten Digitalrechner angesetzt werden kann, bis zum heutigen Tag kontinuierlich gewachsen. Heute geht man aber bereits dazu über, zelluläre Automaten als digitales Universum aufzufassen, das sich nicht nur als Modell komplexer Systeme, sondern um seiner selbst willen zu erforschen lohnt.

Mit „Life 3D“ kann eine schier beliebig große Zahl verschiedener zellulärer Automaten untersucht werden, Geduld und Experimentierfreude vorausgesetzt.

1 ZELLULÄRE AUTOMATEN

1.1 Erklärung und Definition

„Das Schachbrett ist die Welt, die Figuren sind die Phänomene des Universums, und die Spielregeln sind das, was wir die Naturgesetze nennen.“¹

Dieses Zitat von Thomas Henry Huxley faßt alle Elemente eines zellulären Automaten in einem Satz zusammen. Das Schachbrett ist das System, auf dem sich die Evolution der zellulären Automaten abzeichnet, die Figuren sind die jeweiligen Zellgebilde, die der zelluläre Automat hervorbringt, und die Spielregeln sind die Regeln, nach denen sich die Zellen zu richten haben.

Um die Funktionsweise eines zellulären Automaten besser zu verstehen, muß man sich die Filigran-Symmetrie einer Schneeflocke vorstellen. Wie ist es möglich, daß sich die Wassermoleküle immer wieder zu einem anders gestalteten Kunstwerk zusammenlagern? Für die Beschreibung des Wachstumsprozeß von Kristallen oder auch anderer komplexer Systeme der Natur ist der zelluläre Automat das geeignetste Mittel. Am Computer realisierte Automaten sind das entsprechende Computermodell zur Simulation von solchen Prozessen, die mit anderen Modellen nur viel schwieriger und komplizierter beschrieben werden können.

Ein zellulärer Automat ist im Prinzip eine gedankliche Konstruktion bestehend aus einer unendlichen, gitterförmigen Anordnung von Zellen zusammen mit einer endlichen Anzahl von Zuständen, den jede Zelle des Systems einnehmen kann, und einer imaginären Uhr. Jede Zelle befindet sich stets in einem der möglichen Zustände. Mit jedem Ticken der Uhr nimmt jede Zelle einen neuen Zustand ein oder bleibt im alten Zustand, und zwar gewissen Regeln folgend, die mit einer festen Anzahl von Zellen, sogenannten Nachbarn, in Wechselwirkung stehen.² Ein zellulärer Automat ist ein dynamisches diskretes System. Auch die Zeit ist in diskrete Schritte (Generationen) unterteilt.

Folglich sind zelluläre Automaten stilisierte, synthetische Universen. Sie haben ihre eigene Materie, die in ihrem eigenen Raum und in ihrer eigenen Zeit herumwirbelt.³

Die Idee eines zellulären Automaten ist etwa so alt wie die der Digitalrechner selbst. Erste Untersuchungen dazu stellte John von Neumann⁴ (mit maßgeblicher Unterstützung von Stanislaw Ulam) in den frühen fünfziger Jahren an. Mit ihren selbst geschaffenen Regeln,

¹Dewdney, A.K.: Draht- und Teppichwelten. In: Computer Kurzweil 2, Spektrum der Wissenschaft.

²Ebd.

³Dewdney, A.K.: Digitale Krümelmonster. In: Sonderheft 10, Spektrum der Wissenschaft.

⁴Neumann, John von: US-amerikanischer Mathematiker

die einen gleichförmigen zellulären Raum⁵ vorschreiben, versuchten sie selbstreproduzierende Muster zu finden.

1.2 Merkmale

Insgesamt gibt es vier Merkmale, die einen zellulären Automaten charakterisieren.

1.2.1 Geometrie der Zellanordnung

Meistens verwendet man ein rechtwinkliges Kästchen-Gitter aus identischen Quadraten oder in einem dreidimensionalen System aus identischen Würfeln. Natürlich ist jede andere Anordnung, zum Beispiel eine hexagonale (bienenwabenartige) Gitterstruktur für ein Modell des Schneeflockenwachstums, möglich.

1.2.2 Nachbarschaftskonstellation

Das zweite Merkmal ist, welche Plätze in einer bestimmten Anordnung als benachbart zu einer beliebigen Zelle gelten. In zweidimensionalen Systemen wurden bisher hauptsächlich nur zwei Arten von Nachbarschaften verwendet: Neumann- und Moore-Nachbarschaft.

Neumann-Nachbarschaft:

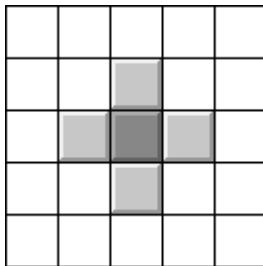


Abb. 1: Benachbarte Zellen (hellgrau) und betrachtete Zelle (dunkelgrau) nach Neumann

John von Neumann verwendete nur die orthogonalen Nachbarn, also die unmittelbar oben, unten, links und rechts von der betrachteten Zelle befindlichen Zellen zur Berechnung des neuen

Moore-Nachbarschaft:

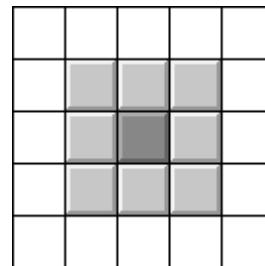


Abb. 2: Benachbarte Zellen (hellgrau) und betrachtete Zelle (dunkelgrau) nach Moore

Edward F. Moore hingegen rechnete auch mit den 4 diagonal benachbarten Zellen.

⁵Die Zustände aller Zellen in einem System unterliegen den gleichen Gesetzen.

Zustands der Zelle.

In einem dreidimensionalen System kann man die Konstellation der Nachbarn noch mehr variieren, weil 26 unmittelbare Nachbarn für die Berechnung des Zustands einer beliebigen Zelle zur Verfügung stehen. Zusätzlich kann man noch den Zustand der betrachteten Zelle selbst zur Berechnung des neuen Zustands hinzuziehen.

1.2.3 Anzahl der möglichen Zellzustände

Die dritte Kenngröße ist die Zahl der Zustände pro Zelle. In meinem Programm kann die Zelle vier Zustände einnehmen. Das erscheint auf dem ersten Blick sehr wenig, aber selbst bei einem binären Automaten⁶ gibt es schon sehr viele Variationsmöglichkeiten, von denen bis heute nur ein Bruchteil wissenschaftlich untersucht werden konnte.

1.2.4 Zellregeln

Erst die Zellregeln, nach denen der Zustand einer Zelle in der nächsten Generation aus der momentanen Nachbarschafts-Konstellation berechnet wird, ermöglichen eine so große Vielfalt im Universum der zellulären Automaten. Unter der Vielzahl möglicher Regelsysteme gibt es ein System, das bei fast allen zellulären Automaten, so auch in meinem Programm, Verwendung findet: *Zähl- oder totalistische Regeln*. Bei solchen Regeln hängt der neue Zustand der Zelle in der nächsten Generation nur von der Zahl, nicht jedoch von der Position der Nachbarn in einem bestimmten Zustand ab. Eine der einfachsten dieser Zählregeln ist die *Paritätsregel*, wonach eine Zelle den Wert 1 erhält, wenn eine ungerade Zahl von Nachbarn diesen Wert hat, oder den Wert 0 im anderen Fall. Eine andere Form von Zählregeln bilden die sogenannten „*Wahl*“-Regeln. Wenn zum Beispiel die Anzahl der Nachbarn in einem bestimmten Zustand einen Schwellenwert überschreitet oder unterschreitet, ändert sich der Zustand der Zelle im Zentrum.

⁶Binärer Automat: Die Zellen eines binären Automaten können nur zwei Zustände einnehmen. (Man kann die Zustände als wahr oder falsch, tot oder lebendig, 0 oder 1, usw. deuten.)

2 LIFE - DAS SPIEL DES LEBENS

In diesem Kapitel werden die bekanntesten Systeme zellulärer Automaten vorgestellt und beschrieben, die auch in „Life 3D“ simuliert werden können.

2.1 John Horton Conways „Game of Life“⁷

Diese wohl bekannteste Welt von zellulären Automaten wurde von dem bekannten Mathematiker John Horton Conway an der Universität Cambridge im Jahre 1968 erfunden und zwei Jahre später von Martin Gardner in seiner Kolumne „Mathematical Games“ im *Scientific American* einem breiten Publikum vorgestellt. So wie seine wuchernden Automaten breitete sich „Game of Life“ auch in den Gehirnen der Leserschaft aus und zog sie in seinen Bann. Dieses zweidimensionale System lohnt die ihm geschenkte Beachtung mit einem Höchstmaß an Vielfalt und Überraschung, obwohl die Zellen dieses Systems nur zwei Zustände (tot oder lebendig) einnehmen können.

In „Life“ werden die Zustände der Zelle mit Hilfe der Moore-Nachbarschaft (also die acht unmittelbaren Nachbarn) und folgenden Zellregeln ermittelt: Eine lebende Zelle überlebt in der nächsten Generation nur dann, wenn sie zwei oder drei lebende Nachbarn hat. Wenn diese Bedingung nicht zutrifft, stirbt sie. Eine tote Zelle wird zum Leben erweckt, wenn sie drei oder mehr lebende Zellnachbarn hat.

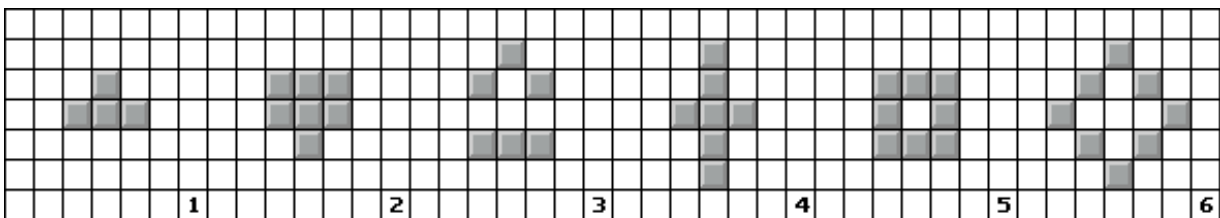


Abb. 3: Die ersten sechs Schritte der Evolution einer Zellanordnung in „Life“ von John Horton Conway

Im Jahre 1969, kurz nachdem er „Life“ erfunden hatte, entdeckte er ein kleines Zellmuster, das sich durch vier Generationen transformiert und am Ende wieder seine Ausgangsgestalt annimmt, jedoch ist es nun um eine Zelle diagonal versetzt. Dieses Phänomen wird als *Gleiter* bezeichnet. Die Verwendung von Gleitern in einer zellulären Welt ist von großer Bedeutung. Mit ihrer Hilfe ist man in der Lage, einen aus Zellen bestehenden Computer zu konstruieren. Statt elektrischer Impulse werden Gleiter für die Übertragung von Signalen verwendet.

⁷vgl. Computer Kurzweil, Spektrum der Wissenschaft, 1988, S. 166, 178, 187, 192.

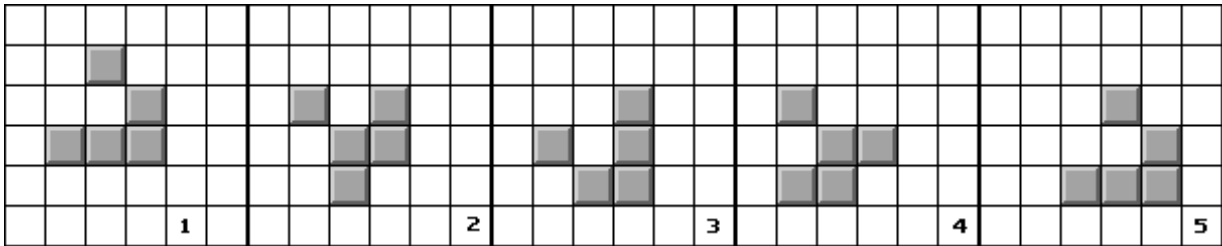
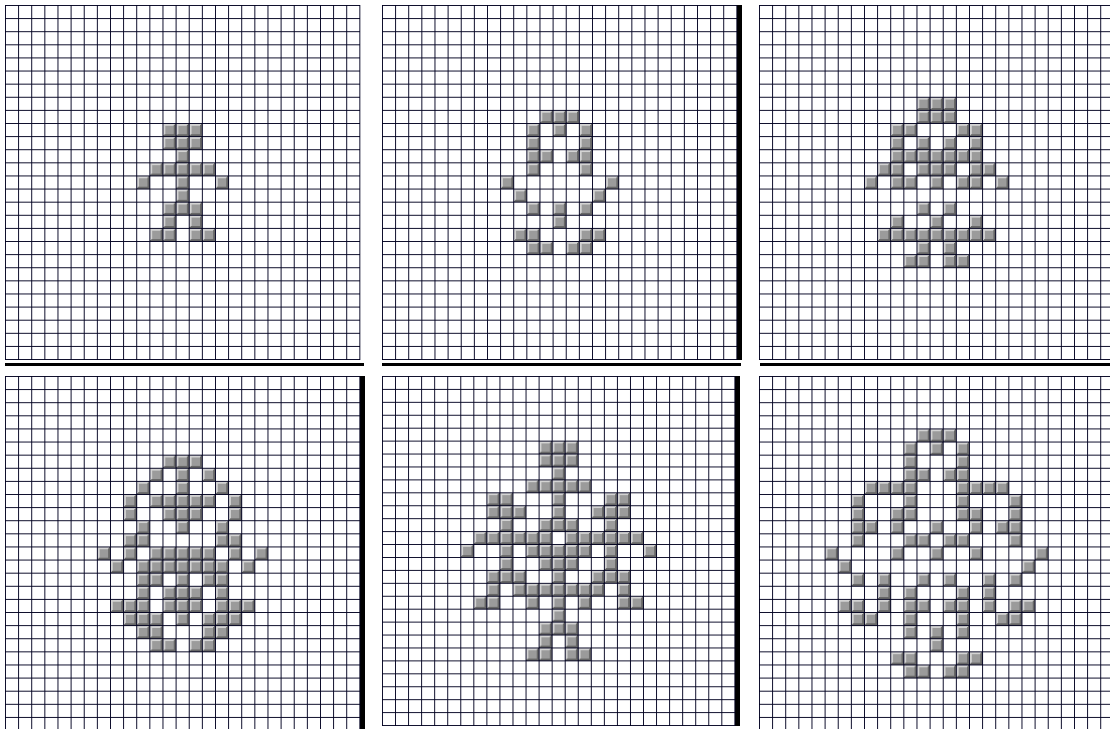


Abb. 4: Transformation eines Gleiters

2.2 Das erste selbstreproduzierende Muster von Edward Fredkin⁸

Im Jahre 1960 entwarf Edward Fredkin vom *Massachusetts Institute of Technology* ein zelluläres System, das in der Lage ist, sich selbst zu replizieren. Schon vor ihm versuchte John von Neumann den Beweis von reproduzierenden Mustern zu erbringen, indem er zeigte, daß ein universeller „Konstrukteur“, der jedes Muster und damit auch sein eigenes Muster erzeugen kann, existiert. Dieser zelluläre Automat besteht jedoch aus ungefähr 200.000 Zellen mit 29 möglichen Zuständen und wurde deshalb wahrscheinlich nie wirklich, weder von Hand noch mit Hilfe eines Computers, konstruiert.



⁸vgl. Computer Kurzweil, Spektrum der Wissenschaft, 1988, S. 164-170.

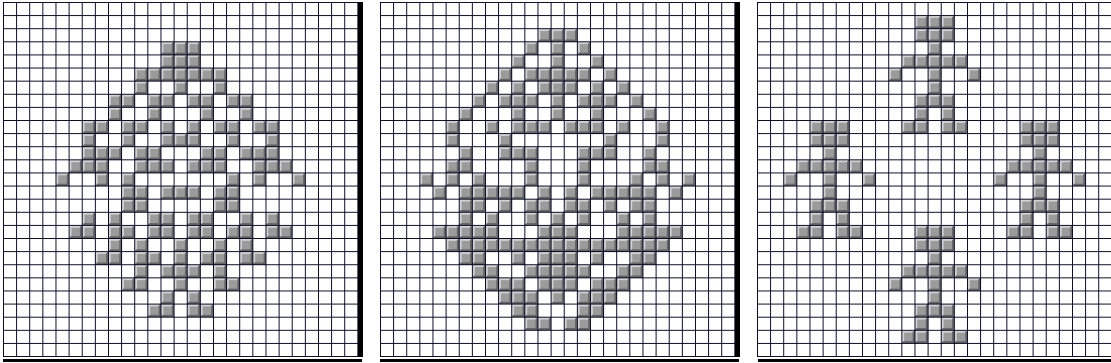


Abb. 5: Die ersten 9 Generationen einer den Fredkinschen Regeln unterworfenen Zellanordnung zur Veranschaulichung eines selbstreproduzierenden Automaten.

Der Nachweis gelang erst Edward Fredkin mit einem viel einfacheren System: Er verwendete die Neumann-Nachbarschaft und die Paritätsregel⁹ zur Bestimmung der Zellzustände. Die Zellmuster in diesem System werden schon nach wenigen Generationen vervierfacht.

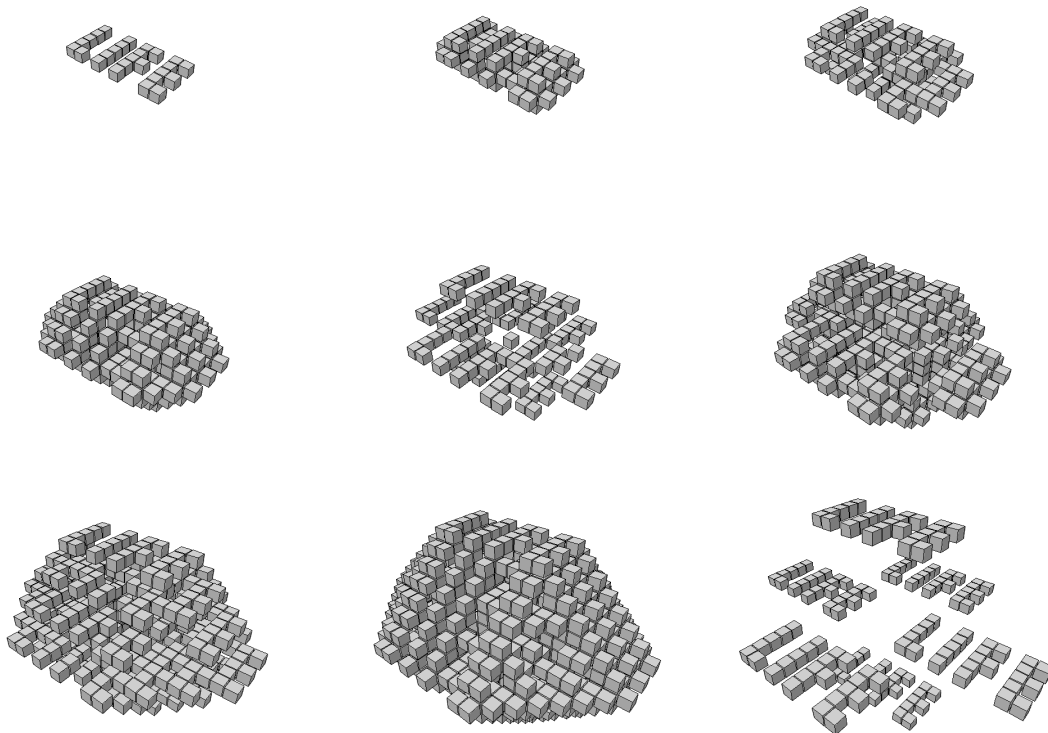


Abb. 6: Die Evolution einer Zellkonfiguration in „Life 3D“ nach den Regeln von Fredkin.

⁹siehe Punkt 1.2.4

Die Zahl der zur Reproduktion benötigten Zyklen hängt von der Komplexität des Ausgangsmusters ab. In „Life 3D“ kann diese berühmte Paritätsregel angewendet werden. Das Ausgangsmuster wird jedoch aufgrund der dreidimensionalen Zellanordnung versechsfacht.

2.3 Carter Bays' Life 4555 und Life 5766¹⁰

Carter Bays, ein Computerwissenschaftler an der Universität von South Carolina, hat nach zahlreichen Versuchen mit dreidimensionalen Systemen diese beiden Versionen gefunden, die sich als würdige Vertreter in drei Dimensionen von Conways „Life“ herausstellten. Die Namen „Life 4555“ und „Life 5766“ basieren auf den im Programm verwendeten Schwellenwerten zur Berechnung des Zustands der Zellen. Die ersten beiden Ziffern entscheiden über das Schicksal der lebenden Zellen, wobei die erste angibt, wieviel lebende Nachbarn mindestens jeweils eine Zelle haben muß, damit sie nicht an „Vereinsamung“ stirbt. Die zweite Ziffer hingegen zeigt an, wieviel lebende Nachbarzellen die Zelle höchstens haben darf, damit sie nicht an „Überbevölkerung“ zugrunde geht. Analog bestimmen die dritte und die vierte Ziffer über das Schicksal der toten Zellen.

Wie in Conways „Life“ gibt es auch hier *Gleiter*, sowie stationäre oder oszillierende Zellanordnungen.

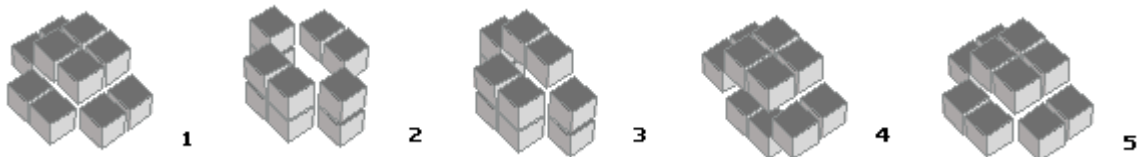


Abb. 7.: Die Transformation eines Gleiters in Life 4555

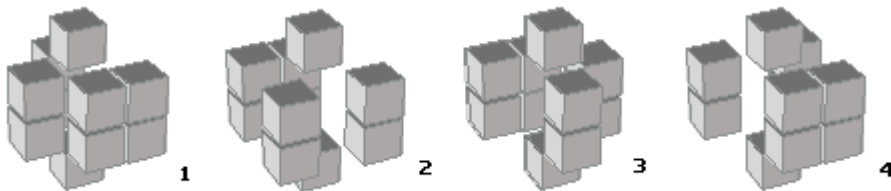


Abb. 8: Oszillierende Zellanordnung names „brockender Bronco“ in Life 4555

¹⁰vgl. Compter Kurzweil, Spektrum der Wissenschaft, 1988, S. 192-197.

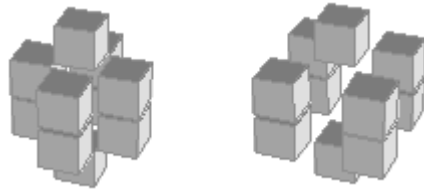


Abb. 9: Noch eine oszillierendes Zellgebilde names „Rotor“



Abb. 10: Einige stationäre Zellkonfigurationen

3 PROGRAMMBESCHREIBUNG

Die nun folgende Beschreibung soll dem Leser einen Überblick über die Fähigkeiten meines Programms „Life 3D“ geben und ihm einen ersten, hoffentlich positiven Eindruck vermitteln. Vielleicht wird im Leser dadurch das Bedürfnis geweckt, sich mit dem Programm zu beschäftigen und mit den zellulären Automaten zu experimentieren.

3.1 Hauptbildschirm

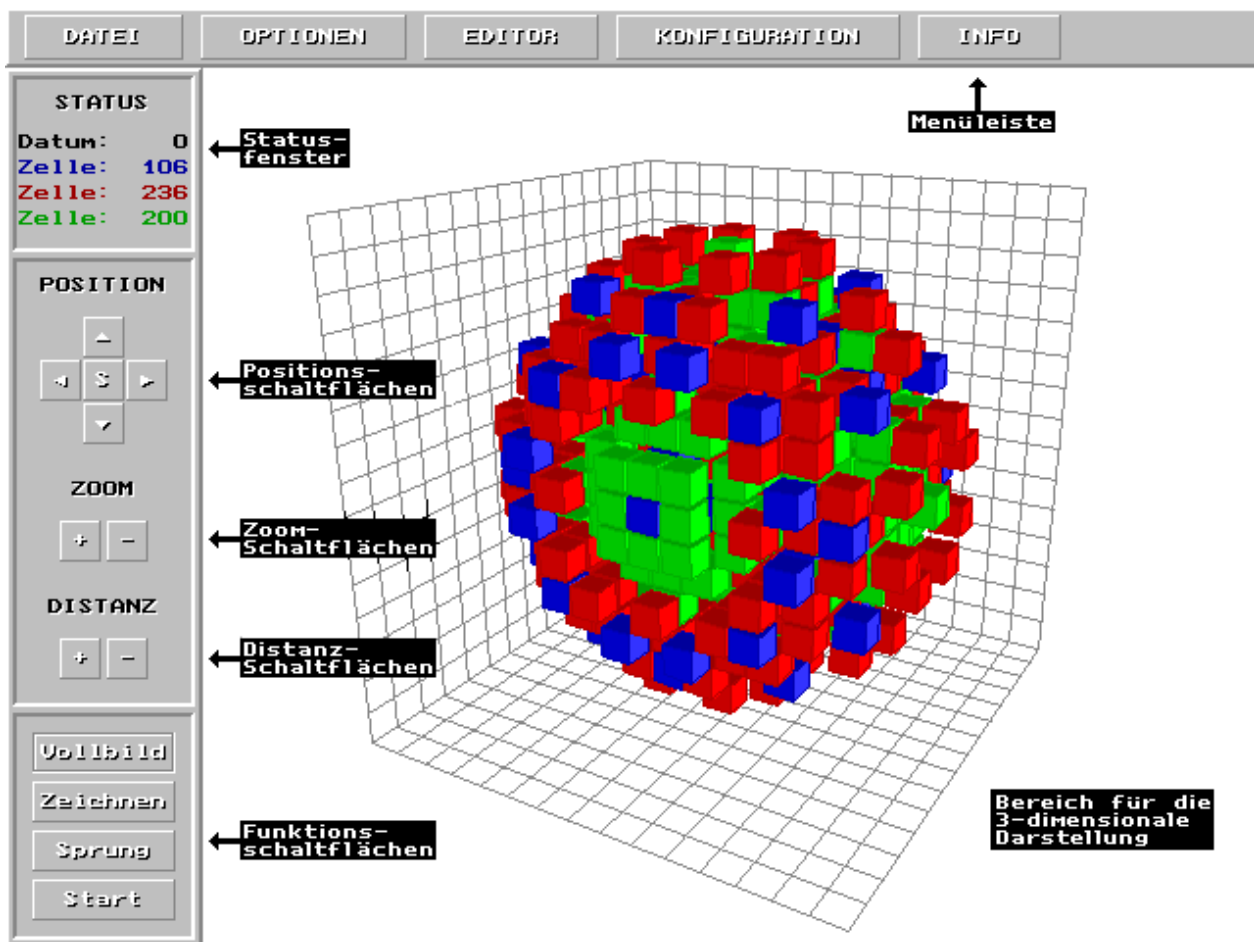


Abb. 11: Hauptbildschirm

Sofort nach dem Start des Programms erscheint der Hauptbildschirm. Der Hauptbildschirm setzt sich aus der Menüleiste am oberen Rand und aus dem am linken Rand des Bildschirms befindlichen Hauptfenster zusammen. Der restliche Teil des Bildschirms wird für die dreidimensionale Darstellung der Zellen verwendet. Zu Beginn des Programms erscheint standardmäßig in diesem Teil lediglich der dreidimensionale Raster.

3.1.1 Hauptfenster

Das Hauptfenster setzt sich aus folgenden Teilen zusammen: das Statusfenster, die Positions-, Zoom-, Distanz- und Funktionsschaltflächen¹¹.

3.1.1.1 Statusfenster

Die Funktion des Statusfensters ist es, dem Benutzer laufend die Anzahl der lebenden Zellen, also die der blauen, roten und grünen Zellen mitzuteilen. Zusätzlich wird das aktuelle Zelldatum angezeigt. Das Statusfenster wird sofort aktualisiert, sobald sich die Anzahl oder Zustände der Zellen im dreidimensionalen Raum ändert (zum Beispiel nachdem man ein Bild geladen hat oder nach der Rückkehr vom Zelleditor in den Hauptbildschirm).

3.1.1.2 Positionsschaltflächen

Mit den Positionsschaltflächen kann man den Standort des Betrachters verändern. Leider ist es nicht möglich, die jeweiligen Zellgebilde von jedem beliebigen Standort aus betrachten, weil die Programmierung von Algorithmen für die dreidimensionale Umsetzung auf den Bildschirm, die eine Betrachtung von allen beliebigen Standorten aus erlauben, den Umfang dieser Fachbereichsarbeit sprengen würde. Aber zumindest kann der Benutzer den Raum um 90° drehen und um 120° kippen.

3.1.1.3 Zoomschaltflächen

Diese Schaltflächen haben lediglich die Aufgabe die dreidimensionale Darstellung der Zellen zu vergrößern oder zu verkleinern.¹²

3.1.1.4 Distanzschaltflächen

Mit den Distanzschaltflächen kann der Benutzer den Effekt der Perspektive entweder verringern oder vergrößern. Standardmäßig ist die kleinste Distanz eingestellt, um den dreidimensionalen Raum möglichst realistisch am Bildschirm wiederzugeben.

¹¹Schaltflächen: Die vom Hintergrund abgehobenen Flächen können mit der Maus betätigt werden und führen eine spezielle Funktion des Programms aus.

¹²Bei maximaler Vergrößerung können jedoch kleine Fehler bei der dreidimensionalen Darstellung der Zellgebilde auftreten.

3.1.1.5 Funktionsschaltflächen

Die Funktionsschaltflächen sind im unteren Teil des Hauptfensters plaziert und setzen sich aus folgenden Schaltflächen zusammen:

- **Vollbild**: Im Unterschied zu den anderen Schaltflächen kann der Benutzer diese Schaltfläche ein- oder ausschalten. Auch das graphische Layout dieser Schaltfläche unterscheidet sich ein wenig von den übrigen. Im gedrückten Zustand ist der Vollbild-Status aktiv und bewirkt, daß die Menüleiste und das Hauptfenster ausgeblendet werden und die dreidimensionale Darstellung am ganzen Bildschirm zu sehen ist. Der Effekt dieser Schaltfläche bezieht sich sowohl auf die drei nachfolgenden Funktionsschaltflächen als auch auf das Laden von gespeicherten Bildern. Um die „Vollbild“-Schaltfläche wieder zu deaktivieren, klickt¹³ man diese einfach mit der linken Maustaste ein zweites Mal an.
- **Zeichnen**: Die „Zeichnen“-Schaltfläche erlaubt dem Benutzer das aktuelle Zellgebilde nach der Rückkehr aus dem Zelleditor oder aus dem Konfigurationseditor wiederaufzubauen. Das 3D-Bild wird nämlich nicht automatisch aktualisiert, weil dadurch der Benutzer ständig Wartezeiten in Kauf nehmen müßte. Wenn die „Vollbild“-Schaltfläche aktiviert ist, erscheint nach Beendigung des Wiederaufbaus in der rechten unteren Ecke des Bildschirms ein kleines Fenster. Das Programm wartet an dieser Stelle, bis die linke oder rechte Maustaste gedrückt wird. Sofort nach dem Drücken einer Maustaste werden die Menüleiste und das Hauptfenster wieder eingeblendet.
- **Start**: Nach dem Anklicken der „Start“-Schaltfläche beginnt das eigentliche „**Spiel des Lebens**“. Gemäß den im Konfigurationseditor eingegebenen Regeln wird die nächste Generation den Zellen berechnet und sofort, je nach Status der „Vollbild“-Schaltfläche, entweder auf dem ganzen Bildschirm oder etwas verkleinert im dafür vorgesehenen Bereich dreidimensional dargestellt. Bei älteren PCs kann der graphische Aufbau der aktuellen Zellgeneration längere Zeit in Anspruch nehmen. Der Benutzer kann während des Zeichenvorgangs mit der linken Maustaste den „Warten“-Status und mit der rechten Maustaste den „Abbruch“-Status verändern:

¹³Klicken: So wird der Vorgang des Drückens und wieder Loslassens der Maustaste bezeichnet.

Warten: Wenn der Benutzer während des Zeichenvorgangs die linke Maustaste drückt, erscheint in der Mitte des Bildschirms ein Fenster mit der Meldung „*Warten aktiviert*“ oder „*Warten deaktiviert*“. „*Warten aktiviert*“ bedeutet, daß das Programm nach jedem graphischen Zellaufbau wartet, bis der Benutzer die linke oder die rechte Maustaste drückt. Wird die linke Taste gedrückt, setzt das Programm mit der Simulation fort und berechnet die nächste Zellgeneration. Nach Drücken der rechten Taste wird die Simulation jedoch abgebrochen. „*Warten deaktiviert*“ bedeutet, daß das Programm ohne Unterbrechung mit der Simulation fortfährt.

Abbruch: Wenn der Benutzer während des Zeichenvorgangs die rechte Maustaste drückt, erscheint in der Mitte des Bildschirms ein Fenster mit der Meldung „*Abbruch aktiviert*“ oder „*Abbruch deaktiviert*“. „*Abbruch aktiviert*“ bedeutet, daß sofort nach Beendigung des Zellaufbaus die Simulation abgebrochen wird. „*Abbruch deaktiviert*“ bedeutet, daß die Simulation nach dem graphischen Zellaufbau mit der Berechnung der nächsten Zellgeneration fortfährt.

- **Sprung**: Die „*Sprung*“-Schaltfläche und die „*Start*“-Schaltfläche haben an und für sich dieselbe Funktion, nur mit dem Unterschied, daß nach dem Anklicken der „*Sprung*“-Schaltfläche die dreidimensionale Darstellung nach jedem Generationswechsel der Zellen unterdrückt wird. Es erscheint lediglich ein Fenster, das den Benutzer den Beginn der Simulation mitteilt und ihn darauf aufmerksam macht, daß er jederzeit die Simulation mit der Escape-Taste abbrechen kann. Während der Simulation wird das Statusfenster im oberen Teil des Hauptfensters ständig aktualisiert, und somit ist der Benutzer in der Lage, den Verlauf des Zellwachstums zu beobachten. Wenn die „*Vollbild*“-Schaltfläche aktiviert ist, erscheint nach Beendigung der Simulation in der rechten unteren Ecke des Bildschirms, analog zur Zeichen-Taste, ein kleines Fenster und das Programm wird erst nach Tastendruck fortgesetzt.

3.1.2 Menüleiste

Die Menüleiste enthält die folgenden Menünamen: **DATEI**, **OPTIONEN**, **EDITOR**, **KONFIGURATION**, **INFO**. Eigentlich sind nur die ersten zwei Einträge der Menüleiste wirkliche Menüs, da diese nach Öffnen per Mausklick eine Liste mit Befehlen zeigen. Wenn man einen Befehlsnamen der Liste wählt, dem keine drei Punkte folgen, wird der Befehl sofort ausgeführt. Wählt man jedoch einen Befehlsnamen, dem drei Punkte folgen, wird ein Dialogfenster angezeigt, in welchem man zusätzliche Spezifikationen festlegen kann, bevor der eigentliche Befehl ausgeführt wird.

3.1.2.1 Datei

3.1.2.1.1 Menübefehl „Neu“

Der Befehl „**Neu**“ bewirkt, daß alle lebenden Zellen in den Zustand 0 (=tot) zurückgesetzt sowie alle Zellregeln gelöscht werden. Weiters wird die Größe des zellulären Raumes auf die Standardgröße (15^3) und das Datum auf 0 zurückgesetzt.

3.1.2.1.2 Menübefehle für Ein- und Ausgabe von Dateien

Nach Aufruf der Befehle „**Bild laden...**“ und „**Konfiguration laden...**“ erscheint ein „**Laden**“-Dialogfenster, nach Aufruf der Befehle „**Bild speichern...**“ und „**Konfiguration speichern...**“ ein „**Speichern**“-Dialogfenster.

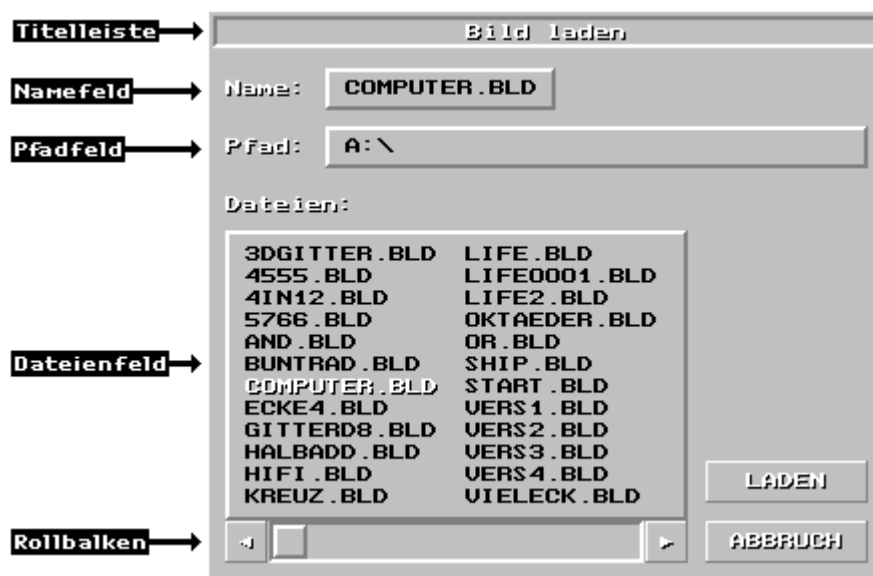


Abb. 11: „Laden“-Dialogfenster

Aufbau des „Laden“- bzw. „Speichern“-Dialogfenster:

- **Titelleiste:** Diese beschreibt die Funktion des Dialogfensters.
- **„Name“-Feld:** Dieses Feld zeigt den momentan ausgewählten Dateinamen an und kann auch angeklickt werden, um den Dateinamen per Tastatur einzugeben (die Erweiterung wird automatisch vergeben).

- **„Pfad“-Feld:** Dieses Feld zeigt den aktuellen Suchpfad an und kann auch angeklickt werden, um den neuen Suchpfad per Tastatur einzugeben.
- **„Dateien“-Feld:** Dieses Feld zeigt das übergeordnete Verzeichnis, alle Unterverzeichnisse und alle Dateien mit der jeweiligen Erweiterung (Erweiterung „BLD“ bei der Ein- und Ausgabe von Bildern, Erweiterung „KFG“ bei der Ein- und Ausgabe von Zellkonfigurationen) im aktuellen Suchpfad an. Man kann direkt in ein neues Verzeichnis, das mit einem Backslash am Ende des Namens gekennzeichnet ist, wechseln, indem man den Verzeichnisnamen mit der rechten Maustaste anklickt. Das Wechseln des Laufwerks ist jedoch auf diese Weise nicht möglich.
- **Rollbalken:** Wenn nicht alle Dateien gleichzeitig anzeigbar sind, dann kann man sich mittels dem Rollbalken, der sich unter dem „Dateien“-Feld befindet, nacheinander die Dateien anzeigen lassen, indem man auf die am linken oder rechten Ende des Rollbalkens befindlichen Schaltflächen klickt.
- **Laden** oder **Speichern**: Nach Anklicken dieser Schaltfläche wird die aktuelle Datei entweder geladen (im „Laden“-Dialogfenster) oder gespeichert (im „Speichern“-Dialogfenster).
- **ABBRUCH**: Will man das Dialogfeld schließen, ohne eine Datei zu laden oder speichern, muß man auf diese Schaltfläche klicken.

Der Benutzer hat nun in diesem Dialogfenster mehrere Möglichkeiten, eine Datei zu laden oder zu speichern.

- **Datei laden:** Der Benutzer kann das Feld **„Name“** anklicken und danach den Namen der zu ladenden Datei per Tastatur eingeben. Wird die Eingabe mit der Return-Taste bestätigt, wird die Datei, sofern sie existiert, sofort geladen, andernfalls erscheint die Meldung *„Datei nicht gefunden“*. Die andere Möglichkeit ist, die zu ladende Datei direkt aus dem Feld **„Dateien“** mit der Maus auszuwählen, wobei diese auch automatisch im Feld **„Name“** angezeigt wird. Man kann die ausgewählte Datei laden, indem man entweder auf die Schaltfläche „Laden“ oder direkt mit der rechten Maustaste auf die zu ladende Datei klickt.

- **Datei sichern:** Das Programm schlägt automatisch einen Namen für die zu speichernde Datei vor (z.B. LIFE0001.BLD). Will der Benutzer jedoch selbst einen Namen vergeben, muß er auf das Feld „**Name**“ klicken, den neuen Namen eingeben und mit der Return-Taste bestätigen. Sind jedoch im vom Benutzer vergebenen Namen ungültige Zeichen vorhanden, erscheint die Meldung „*Datei kann nicht erstellt werden*“. Existiert die Datei bereits, erscheint ein Dialogfenster mit der Frage, ob die bereits bestehende Datei überschrieben werden soll. Man kann auch eine bereits existierende Datei im Feld „**Dateien**“ auswählen und diese überschreiben, indem man entweder auf die Schaltfläche „*Speichern*“ oder direkt mit der rechten Maustaste auf die zu überschreibende Datei klickt.

3.1.2.1.3 Menübefehl „Programm beenden...“

Dieser Befehl wird nicht unmittelbar ausgeführt, sondern es erscheint ein Dialogfenster, das den Benutzer auffordert, seine Wahl zu bestätigen. Nach Bestätigung des Befehls, wird „**LIFE 3D**“ beendet und vollständig aus dem Speicher entfernt.

3.1.2.2 Optionen

Das „**OPTIONEN**“ Menü beinhaltet eine Liste von Befehlen, die dem Benutzer erlauben, einerseits die graphische Darstellung der Zellen seinem Geschmack anzupassen, und andererseits grundlegende Regeln für die Berechnung der nächsten Zellgeneration zu verändern.

3.1.2.2.1 Menübefehl „3D-Raster“

Mit diesem Befehl kann der Benutzer die Darstellung des dreidimensionalen Rasters während der Simulation ein- oder ausschalten. Der aktuelle Status wird neben dem Befehl im Menü angezeigt.

3.1.2.2.2 Menübefehl „Zellrahmen“

Dieser Befehl bewirkt im aktivierten (eingeschalteten) Zustand, daß die Zellen mit einem Rahmen dargestellt werden, um so einen besseren Kontrast zu erzielen. Im deaktivierten Zustand wird die Darstellung des Rahmens

unterdrückt. Der aktuelle Status wird ebenfalls neben dem Befehl im Menü angezeigt.

3.1.2.2.3 Menübefehl „Zelluniversum“

Im Programm kann man zwei verschiedene Zelluniversen benutzen, das „*unendliche*“ und das „*endliche Universum*“. Das aktuelle Universum wird neben dem Befehl im Menü angezeigt.

- **Unendliches Universum:** Der Idealfall wäre, die Evolution der zellulären Automaten in einem unendlich großen Raum zu simulieren. Doch es ist unmöglich, mit dem Computer einen unendlich großen Raum (d.h. ein unendlich großes Datenfeld) zu generieren. Außerdem ist es nicht sehr sinnvoll, dieses dreidimensionale Datenfeld zu groß zu wählen, da erstens die dreidimensionale Darstellung am Bildschirm sehr unübersichtlich und zweitens die Geduld des Benutzers unnötig strapaziert werden würde. (Die zum Schaffen und Zeichnen einer neuen Generation benötigte Zeit wächst proportional zur Zahl der beteiligten Zellen.) Daher „verschwinden“ einfach alle lebenden Zellen, die über den vorgegebenen Raum (maximal 28^3 Zellen) hinauswandern. Diese Zellen werden in der darauffolgenden Generation nicht mehr am Bildschirm dargestellt, jedoch weiterhin im Datenfeld behalten, damit die Berechnung der folgenden Zellgenerationen nicht allzu sehr verfälscht wird. Erst nach 2 Generationen werden diese endgültig von den nachrückenden Zellen aus dem Datenfeld verdrängt.
- **Endliches Universum:** Im Gegensatz zum unendlichen Universum bleibt dieses Universum in sich geschlossen und kompakt. Wenn eine Zelle am Rand des Universums angelangt ist, wird sie nicht aus dem Datenfeld gestoßen, so wie es beim „*unendlichen Universum*“ der Fall ist, sondern erscheint wieder auf der gegenüberliegenden Seite. Diese Regelung gilt für alle drei Richtungen. Damit werden die Zellen an den gegenüberliegenden Flächen für benachbart erklärt, sofern sie sich in derselben Ebene befinden. Das ist die einzige Möglichkeit, ein endlich ausgedehntes, aber unbegrenztes¹⁴ dreidimensionales Feld zu erzeugen. Somit kann keine lebende Zelle jemals dieses Miniuniversum verlassen.

3.1.2.2.4 Menübefehl „Zellabstand einstellen...“

¹⁴vgl. Computer Kurzweil, Spektrum der Wissenschaft, 1988, S. 168.

Auch dieser Befehl wird nicht sofort ausgeführt, sondern es erscheint ein Dialogfenster, das dem Benutzer erlaubt, den Abstand (maximal 6 Pixel) der Zellen voneinander einzustellen. Standardmäßig ist der Abstand ein Pixel groß. Um jedoch das Innenleben mancher dreidimensionaler Zellgebilde besser beobachten zu können, ist es ratsam, den Abstand zu vergrößern.

3.1.2.2.5 Menübefehl „Nachbarn auswählen...“

Die im Befehl enthaltenen drei Punkte künden schon an, daß nach Anklicken dieses Befehls ein Dialogfenster erscheint. Dieses bietet dem Benutzer die Möglichkeit, die Nachbarn, die für die Berechnung des neuen Zustands jeder einzelnen Zelle herangezogen werden, zu konfigurieren. Dadurch ergibt sich eine Fülle von Variationsmöglichkeiten, mit denen der Benutzer experimentieren kann.



Abb. 12: „Zellnachbarn“-Dialogfenster

Aufbau des „Zellnachbarn“-Dialogfenster:

- **Titelleiste:** Diese enthält lediglich den Namen „Zellnachbarn“.
- **Auswählbare Elemente mit Kontrollkästchen:** Es gibt 4 verschiedene Elemente, die der Benutzer aktivieren kann, indem er mit der Maus auf das entsprechende Kontrollkästchen klickt. Im aktivierten Zustand enthält das Kontrollkästchen einen schwarzen Kreis. Durch nochmaliges Klicken auf das Kontrollkästchen wird das Element wieder deaktiviert. Die Elemente





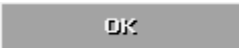
sind die verschiedene Arten von Nachbarn, die der Benutzer auswählen kann, um eine individuelle Nachbarschaft zusammenzustellen:

Flächennachbarn: Die Zellen, die an die Flächen der betrachteten Zelle grenzen, werden Flächennachbarn genannt.

Kantennachbarn: Diese Zellen grenzen an die Kanten der betrachteten Zelle.

Ecknachbarn: Diese Zellen grenzen an die Ecken der Zelle im Zentrum.

Zelle im Zentrum: Schließlich hat der Benutzer die Möglichkeit, den Zustand der betrachteten Zelle selbst für die Berechnung des Zustands in der nächsten Generation miteinzubeziehen.

- **Dreidimensionale Darstellung der aktivierten Nachbarn:** Durch diese Darstellung wird dem Benutzer deutlich vor Augen geführt, welche Nachbarn aktiviert sind. Zusätzlich werden die verschiedenen Arten von Nachbarn in verschiedenen Farben dargestellt.
- **„Reihenfolge“-Schaltflächen:** Mit Hilfe dieser Schaltflächen kann der Benutzer verschiedene Nachbarschaften zusammenstellen, die vom Programm kontinuierlich nacheinander für die Berechnung der nächsten Generation verwendet werden. Mit  bzw.  kann man die Anzahl (maximal 4) der verschiedenen Nachbarschaften bestimmen. Rechts von diesen, erscheint für jede weitere Nachbarschaft eine Schaltfläche (). Mit diesen Schaltflächen kann der Benutzer jede Nachbarschaft anwählen und dann durch Anklicken der vorher beschriebenen Elemente konfigurieren.
- : Durch Anklicken dieser Schaltfläche werden alle Werte auf die Standardwerte zurückgesetzt. (Standardmäßig gibt es nur eine Nachbarschaft, die sich aus den Flächen-, Kanten- und Ecknachbarn zusammensetzt.)
- : Um dieses Dialogfenster zu schließen und die aktuellen Einstellungen zu akzeptieren, klickt man auf diese Schaltfläche.

3.1.2.2.6 Menübefehl „Datum auf Null setzen“

Dieser Befehl bewirkt, daß das Zelldatum auf Null zurückgesetzt wird.

3.1.2.3 Editor

Nach Anklicken von „**EDITOR**“ in der Menüleiste, wird kein Menü geöffnet, sondern man wechselt sofort in den Zelleditor (siehe Punkt 3.2), der den ganzen Bildschirm in Anspruch nimmt.

3.1.2.4 Konfiguration

Nach Anklicken von „**KONFIGURATION**“ in der Menüleiste gelangt man in den Konfigurationseditor (siehe Punkt 3.3), der ebenfalls wie der Zelleditor den ganzen Bildschirm in Anspruch nimmt.

3.1.2.5 Info

Nach Anklicken von „**INFO**“ erscheint ein Fenster, das Informationen über das Programm und den Autor liefert.

3.2 Zelleditor

Mit dem Zelleditor ist der Benutzer in der Lage, seine eigenen Gebilde aus Zellen zusammenzustellen, mit denen er experimentieren kann. Per Mausklick können die Zustände der einzelne Zellen vom Benutzer verändert werden, d.h. er kann alle Arten von Zellen beliebig im dreidimensionalen Raum plazieren. Zur Verfügung stehen blaue, rote, grüne und tote Zellen (diese werden zum Löschen der anderen verwendet). Der Editor ermöglicht dem Benutzer eine bequeme und benutzerfreundliche Gestaltung des zellulären Universums.

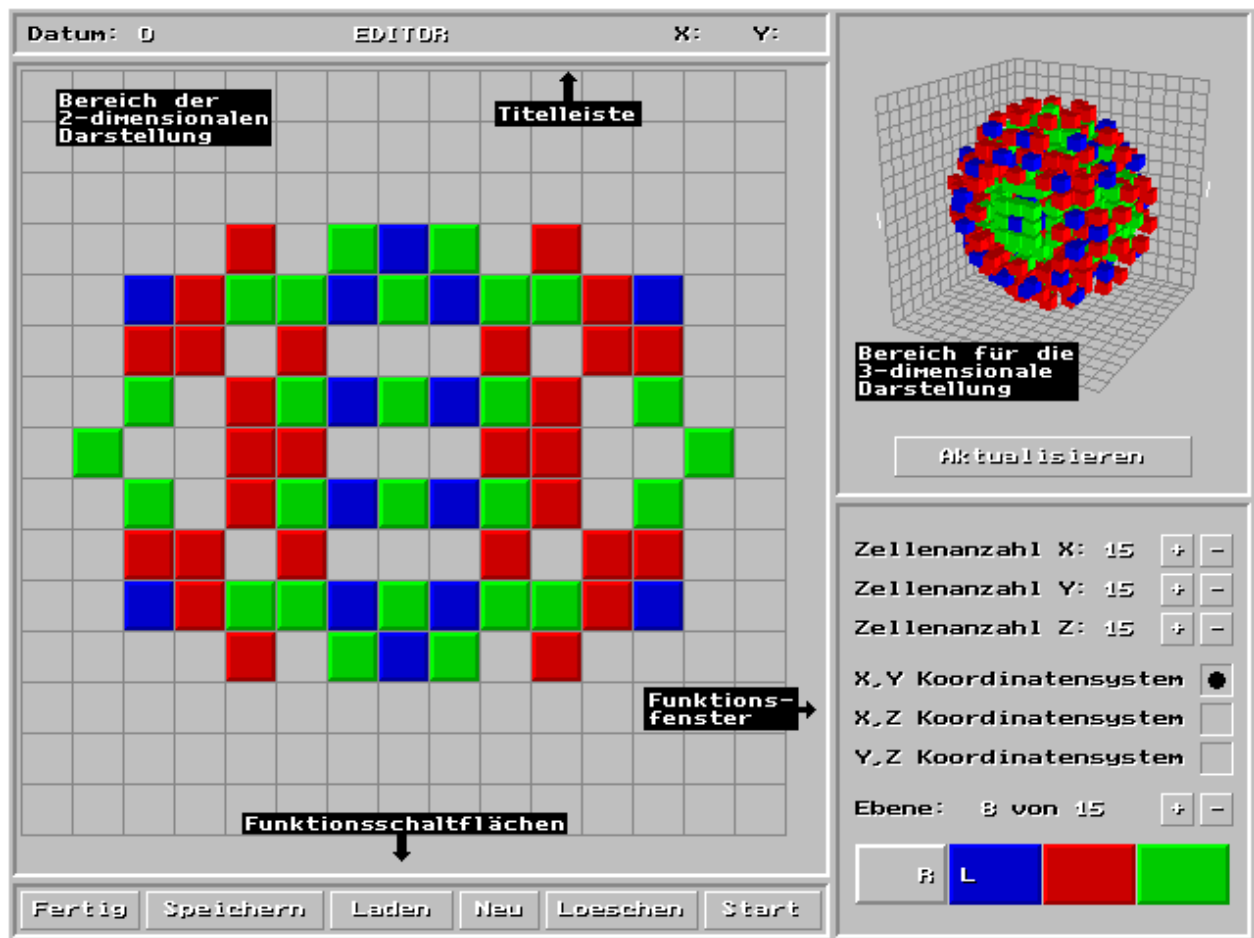


Abb. 13: Zelleditor

3.2.1 Titelleiste



Die Titelleiste zeigt das aktuelle Zelldatum, den Titel „EDITOR“ und die Mausposition, sofern sich der Mauszeiger über der Bearbeitungsebene, also im Bereich für die zweidimensionale Darstellung befindet.



3.2.2 Funktionsschaltflächen

- **Fertig**: Nach Beendigung des Editiervorgangs verläßt man den Editor und kehrt in den Hauptbildschirm zurück, indem man auf diese Schaltfläche klickt.
- **Laden**: Nach Anklicken dieser Schaltfläche erscheint das bereits unter 3.1.2.1.2 beschriebene „Laden“-Dialogfenster. Somit hat der Benutzer die Möglichkeit, seine bereits gespeicherten zellulären Gebilde auch direkt im Editor zu laden.
- **Speichern**: Nach Anklicken dieser Schaltfläche erscheint das unter 3.1.2.1.2 beschriebene „Speichern“-Dialogfenster. Der Benutzer kann somit seine selbst gestalteten Zellkonfigurationen auch im Editor abspeichern.
- **Neu**: Dieser Befehl bewirkt, daß alle Zellen im Raum gelöscht werden, d.h. die Zustände aller Zellen werden auf 0 (=tot) zurückgesetzt.
- **Loeschen**: Mit diesem Befehl kann man alle Zellen der aktuellen Ebene löschen.
- **Start**: Nach Anklicken dieser Schaltfläche beginnt die Simulation. Im Unterschied zur dreidimensionalen Darstellung im Hauptbildschirm hat der Benutzer hier die Möglichkeit, das Zellwachstum in jeder beliebigen Ebene zu betrachten. Durch Drücken der linken Maustaste kann die Simulation jederzeit abgebrochen werden. Die kleine dreidimensionale Darstellung in der rechten oberen Ecke des Bildschirms wird während der Simulation aus Zeitgründen nicht nach jeder Generation aktualisiert.

3.2.3 Funktionsfenster

Das Funktionsfenster setzt sich aus folgenden Elementen zusammen:

- **Schaltflächen zur Einstellung der Größe des Raumes**: Mit  und , die dreimal (also für jede Dimension) vorhanden sind, kann man die Größe des zellulären Universums in allen drei Richtungen festlegen. Die aktuelle Einstellung wird links von den Schaltflächen angezeigt.

- **Auswahl zum Festlegen der Stellung der Bearbeitungsebene:** Durch Anklicken der Auswahlkästchen kann die jeweilige Stellung der Ebene im dreidimensionalen Raum aktiviert werden.
- **Schaltflächen, um die Ebene zu wechseln:** Mit  und  kann der Benutzer die Ebene wechseln und somit den dreidimensionalen Raum schichtweise bearbeiten. Die aktuelle Ebene und die Ebenenanzahl wird links von den Schaltflächen angezeigt.
- **Auswahl der möglichen Zustände der Zellen:** Durch Anklicken der farbigen Flächen mit der linken bzw. rechten Maustaste werden beide Maustasten mit der entsprechenden Farbe, in welcher die Zellen dann bei der Editierung in der Bearbeitungsebene dargestellt werden, belegt. Die aktuelle Maustastenbelegung wird durch ein „L“ (für die linke Maustaste) und ein „R“ (sinngemäß für die rechte Maustaste) auf den Farbflächen wiedergegeben. Die Farben sind die möglichen Zustände, die eine Zelle annehmen kann.

3.2.4 Bereich für die zweidimensionale Darstellung

Hier wird die aktuelle Ebene des Raumes angezeigt, die mit der Maus bearbeitet werden kann. Durch Drücken der linken oder rechten Maustaste wird an der aktuellen Position, die in der Titelleiste angezeigt wird, eine Zelle in dem vom Benutzer gewählten Zustand gesetzt.

3.2.5 Bereich für die dreidimensionale Darstellung

In diesem Bereich wird das ganze Zellgebilde dreidimensional dargestellt, damit der Benutzer nicht die Übersicht verliert. Ebenso wird die aktuelle Ebene mit zwei weißen Balken am Rand des dreidimensionalen Raster angezeigt. Die dreidimensionale Darstellung wird nicht automatisch nach jeder Veränderung der Zellzustände aktualisiert. Der Benutzer kann sich aber durch Anklicken der „Aktualisieren“-Schaltfläche, die sich unter der dreidimensionalen Darstellung befindet, jederzeit seine selbstkreierten Zellgebilde dreidimensional anzeigen lassen, um die Richtigkeit der Zellanordnung zu überprüfen.

3.3 Konfigurationseditor

Der Konfigurationseditor ist neben dem Zelleditor der zweite große Teil dieses Programms. Mit dessen Hilfe können die Regeln des „Lebens“, nach denen sich die Zellen zu richten haben, geschaffen werden. Der Konfigurationseditor bietet dem Benutzer durch seine integrierten Funktionen eine große Anzahl an Variationsmöglichkeiten.

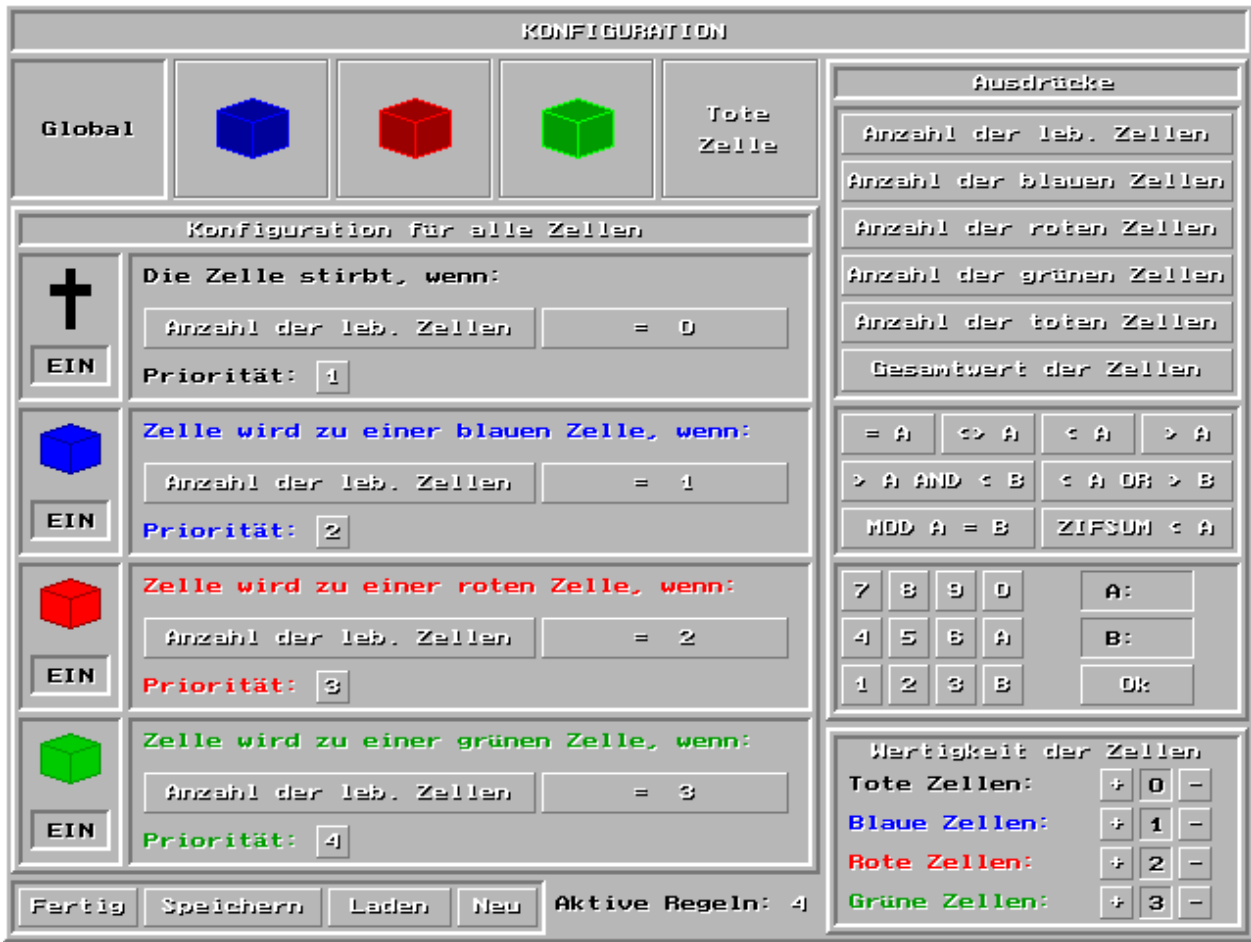


Abb. 14: Konfigurationseditor

3.3.1 Titelleiste

Diese enthält lediglich die Überschrift „KONFIGURATION“.

3.3.2 Zellschaltflächen

Mit diesen etwas größer gehaltenen Schaltflächen kann man das Zellregelfenster für jede einzelne Zelle, sowie ein Regelfenster, das Regeln für alle Zellen beinhaltet, anwählen.

3.3.3 Zellregelfenster

Dieses Fenster bietet dem Benutzer eine Übersicht der bestehenden Regeln für eine bestimmte Zelle. Auf der linken Seite befindet sich ein Symbol für jede Zelle, in welche sich die mittels der Zellschaltflächen ausgewählte Zelle „verwandeln“ kann. Unter dem Symbol befinden sich die *„Ein/Aus“-Schaltflächen*. Im freien Bereich rechts von den Symbolen werden die vom Benutzer definierten Zellregeln dargestellt. Ein besonderes Merkmal stellen die *Prioritätsschaltflächen* dar, die eine genauere Definition der Zellregeln ermöglichen.

„Ein/Aus“-Schaltflächen: Im gedrückten Zustand sind diese Schaltflächen eingeschaltet und zeigen an, daß die rechts davon befindliche Zellregel aktiv ist. Um aktivierte Zellregeln zu löschen, klickt man auf diese Schaltfläche.

Prioritätsschaltflächen: Diese Schaltflächen befinden sich direkt unter dem Bereich für die Zellregeln und sind mit den Nummern 1 bis 4 beschriftet. Um die Prioritätsverhältnisse der bestehenden Zellregeln zu verändern, klickt man auf diese Schaltflächen und verändert somit die Anordnung der Symbole am linken Rand des Fensters. Prioritätskonflikte liegen dann vor, wenn sich Zellregeln des neu zu berechnenden Zustands überlappen und somit keine eindeutige Zustandszuordnung eines Zelltyps vorliegt. Standardmäßig besitzt die Zellregel, die das Sterben einer Zelle beschreibt, die Priorität 1 (= höchste Priorität), Regeln für eine „Metamorphose“ in eine blaue Zelle besitzen die Priorität 2, die „Verwandlung“ in rote Zellen Priorität 3 und in grüne Zellen Priorität 4 (= niedrigste Priorität).

3.3.4 Funktionsschaltflächen

Wie im Zelleditor befinden sich auch hier Schaltflächen, die unter anderem auch das Laden und Speichern von Zellregelkonfigurationen im Konfigurationseditor erlauben.

- **Fertig**: Nach Fertigstellung der Zellregeln klickt man auf diese Schaltfläche, um den Konfigurationseditor zu verlassen und in den Hauptbildschirm zurückzukehren.
- **Laden**: Nach Anklicken dieser Schaltfläche erscheint das bereits unter 3.1.2.1.2 beschriebene Dialogfenster. Der Benutzer hat somit die Möglichkeit, Zellregelkonfigurationen im Konfigurationseditor zu laden.
- **Speichern**: Nach Anklicken dieser Schaltfläche erscheint das Speichern-Dialogfenster und der Benutzer kann seine selbstdefinierten Zellregeln sofort abspeichern.
- **Neu**: Dieser Befehl bewirkt, daß alle bestehenden Zellregeln gelöscht werden.

3.3.5 Integrierte Funktionen

Der Benutzer kann seine Zellregeln mit Hilfe dieser von mir vorgegebenen Funktionen zusammenstellen. Diese Funktionen sind in folgende drei Teile geteilt:

- **Ausdrücke der zu berücksichtigenden Nachbarn:**

Anzahl der leb. Zellen: Die blauen, roten und grünen Zellen werden berücksichtigt.

Anzahl der blauen Zellen: Nur die blauen Zellen werden berücksichtigt.

Anzahl der roten Zellen: Nur die roten Zellen werden berücksichtigt.

Anzahl der grünen Zellen: Nur die grünen Zellen werden berücksichtigt.

Anzahl der toten Zellen: Nur die toten Zellen werden berücksichtigt.

Gesamtwert der Zellen: Dieser Ausdruck bildet eine Ausnahme. Hierbei werden die Werte, die im Wertigkeitsfenster vergeben werden können, von allen Nachbarn addiert.

- **Mathematischen Ausdrücke:**

= A: gleich A

<> A: ungleich A

> A : größer als A

< A : kleiner als A

> A AND < B : größer als A und kleiner als B

< A OR > B : kleiner als A oder größer als B

MOD A = B : Hierbei wird der Ausdruck vom 1. Teil durch A dividiert und wenn der Restbetrag dieser Division gleich B ist, ist die Zellregel erfüllt.

ZIFSUM < A : Wenn die Ziffernsumme des Ausdrucks vom 1. Teil gleich A ist, wird die Zellregel erfüllt.

- **Wertzuweisung der Variablen:**

In diesem Abschnitt werden die Werte für die Variablen A und B durch Anklicken der Zahlen vergeben.



Erstellen von Zellregeln: Das Erstellen von Zellregeln ist absichtlich einfach und übersichtlich gehalten und erfolgt ausschließlich per Maussteuerung, damit auch Unerfahrene sich nicht sofort frustriert und gelangweilt von diesem Programm abwenden. Man muß natürlich schon über die Funktionsweise von zellulären Automaten, die schon zu Beginn erklärt wurden, informiert sein, um wirklich brauchbare zelluläre Automaten, also solche, die sich nicht ewig in die Unendlichkeit ausbreiten, zu schaffen. Natürlich ist auch viel Geduld und Freude am Experimentieren notwendig. Der Faktor des Zufalls spielt erfahrungsgemäß eine wesentliche Rolle, denn die Regeln können eine derartige Komplexität annehmen, daß es beinahe unmöglich ist, die nächsten Generationen auch von nur wenigen Zellen vorauszuahnen.

Die Regeln können aus den vorher beschriebenen Ausdrücken zusammengesetzt werden. Es können maximal 20 Regeln aufgestellt werden: Regeln für die vier verschiedenen Zelltypen, einschließlich einem globalen Regelsystem, das für alle Zellzustände gleichermaßen gültig sind. Es ist jedoch zu beachten, daß globale Regeln die niedrigste Priorität besitzen. Das bedeutet, daß der Benutzer die Möglichkeit hat, allgemeine Regeln für alle Zelltypen aufzustellen, jedoch für einzelne Zellarten bei einer speziellen Zellanordnung Ausnahmen definieren kann.

Um jetzt eine Zellregel zu definieren, klickt man auf die „Ein/Aus“-Schaltfläche und der Mauszeiger wird in den Bereich der integrierten Funktionen versetzt, wo der Benutzer nacheinander einen Ausdruck auswählen und dann die Werte für die Variablen A und B vergeben muß. Die Zellregel wird durch die „Ok“-Schaltfläche bestätigt, worauf sie sofort in dem dafür vorgesehenen Bereich des Zellregelfensters

erscheint. Natürlich können die Regeln auch nachträglich modifiziert werden, indem man auf eine der zwei Teile, aus denen die Zellregel besteht, klickt und einen neuen Ausdruck auswählt bzw. neue Werte vergibt.

3.3.6 Wertigkeitsfenster

In diesem Fenster kann man jeder Zelle einen bestimmten Wert von 0 bis 9 zuweisen. Mit  und  kann der Wert verändert werden. Diese Werte werden erst dann berücksichtigt, wenn von den integrierten Funktionen der Ausdruck „*Gesamtwert der Zellen*“ ausgewählt wird. Den Gesamtwert der Zellen erhält man durch Addition der Werte von allen aktivierten Nachbarn. Dadurch kann man Regeln erstellen, die nur bei einer speziellen Zellanordnung zutreffen.

ANHANG A

QUELLTEXT PROGRAMM „LIFE 3D“

```
PROGRAM LIFE3D;
```

```
USES CRT,GRAPH,DOS,MAUSMENU,FASTTEXT;
```

```
Const Distanz:Word = 600;  
      MaxZellAnzahl_X = 28;  
      MaxZellAnzahl_Y = 28;  
      MaxZellAnzahl_Z = 28;  
      DrehWinkel:Integer = -30;  
      KippWinkel:Integer = 20;  
      Zoom_Faktor:Integer = 140;  
      sp:shortint = 1;  
      Datum:Integer = 0;  
      Header1:Byte = 255;  
      erwpic = 'BLD';  
      erwbed = 'KFG';  
      mtext:Array[1..5] of String = ('DATEI','OPTIONEN','EDITOR','KONFIGURATION','INFO');
```

```
Type Feld3D = Array[0..MaxZellAnzahl_X+1,0..MaxZellAnzahl_Y+1,0..MaxZellAnzahl_Z+1] Of Byte;  
      ConfigTyp = Record  
      CellSwrch:Boolean;  
      exp1:Byte;  
      exp2:Byte;  
      exp3:Array[1..2] of Byte;  
      prior:Byte;  
      wert:Byte;  
      End;
```

```
Var CosDW, SinDW, DW, CosKW, SinKW, KW: Real;  
    T: Array[1..9] Of Real;  
    ux, uy, i, j, k, kl, ZellFaktor, MaxX, MaxY: Integer;  
    Feld, NeuFeld: Feld3D;  
    SeqAnz, ZellenX, ZellenY, ZellenZ: Byte;  
    reg: Registers;  
    EndOfProg, ZellCheckZ, ZellCheckY, DrawStop, MausKlick: Boolean;  
    FullScreenChk, RasterChk, RahmenChk, FeldTypChk: Boolean;  
    nb: Array[1..4,1..4] of Boolean;  
    cvar: Array[1..5,1..4] of ConfigTyp;  
    Size1, Size2: Word;  
    P1, P2, PChars: Pointer;  
    Mnu: Menu;  
    M: Maus;  
    mx, my, mt: Integer;
```

```
{-----Grafikinitialisierung-----}  
Procedure Grafik_Ein;  
Const Bgipath="";  
Var gd,gm,error: Integer;  
Begin
```

```

gd:=VGA;
gm:=VGAHI;
InitGraph(gd,gm,bgipath);
error:=GraphResult;
If error<>0 Then Begin
  Writeln;
  Writeln('Error: '+GraphErrorMsg(error));
  Halt;
End;
End;
{----Initialisierung der Unit FastText----}
Procedure FastTextInit;
Begin
  SetVga;
  GetMem(pchars,256*8);
  SetCharMem(pchars^);
  InitFast;
End;
{----Neudefinition der Farben----}
Procedure Farben_def;
Const Colors: Array[0..maxcolors] Of Word = (0,1,2,3,4,5,20,7,56,57,58,59,60,61,62,63);
Begin
  SetRGBpalette(colors[0],0,0,0);
  SetRGBpalette(colors[1],0,0,30);
  SetRGBpalette(colors[2],0,0,40);
  SetRGBpalette(colors[3],0,0,50);
  SetRGBpalette(colors[4],30,0,0);
  SetRGBpalette(colors[5],40,0,0);
  SetRGBpalette(colors[6],50,0,0);
  SetRGBpalette(colors[7],0,30,0);
  SetRGBpalette(colors[8],0,40,0);
  SetRGBpalette(colors[9],0,50,0);
  SetRGBpalette(colors[10],0,0,35);
  SetRGBpalette(colors[11],35,0,0);
  SetRGBpalette(colors[12],0,35,0);
  SetRGBpalette(colors[13],20,20,20);
  SetRGBpalette(colors[14],40,40,40);
  SetRGBpalette(colors[15],60,60,60);
End;
{----Prozedur für die Textausgabe----}
Procedure SetText(x,y:Integer;str:String;c1,c2:Byte);
Begin
  SetColor(c2);
  ShowTextXY(x+1,y+1,str);
  SetColor(c1);
  ShowTextXY(x,y,str);
End;
{----Initialisierung (Zeichnen) des Hauptbildschirms----}
Procedure Screen_Init;
Begin
  Mnu.Win(0,0,maxX,30,15,14,13);  Mnu.Win(0,31,100,maxY,15,14,13);
  Mnu.Win(4,35,96,124,13,14,15);  Mnu.Win(4,128,96,356,13,14,15);
  Mnu.Win(4,360,96,maxY-4,13,14,15); Mnu.ButtonOut(28,322,48,342, '+');

```



```

Mnu.ButtonOut(52,322,72,342,'-'); Mnu.ButtonOut(40,158,60,178,#30);
Mnu.ButtonOut(40,202,60,222,#31); Mnu.ButtonOut(18,180,38,200,#17);
Mnu.ButtonOut(62,180,82,200,#16); Mnu.ButtonOut(40,180,60,200,'S');
Mnu.ButtonOut(28,262,48,282,'+'); Mnu.ButtonOut(52,262,72,282,'-');
SetText((100-textwidth('DISTANZ')) div 2,302,'DISTANZ',0,15);
SetText((100-textwidth('POSITION')) div 2,138,'POSITION',0,15);
SetText((100-textwidth('ZOOM')) div 2,242,'ZOOM',0,15);
SetText((100-textwidth('STATUS')) div 2,45,'STATUS',0,15);
SetColor(0); ShowTextXY(7,65,'Datum:');
SetColor(1); ShowTextXY(7,78,'Zelle:');
SetColor(4); ShowTextXY(7,91,'Zelle:');
SetColor(7); ShowTextXY(7,104,'Zelle:');
Mnu.ButtonOut(10,4,90,26,mtext[1]); Mnu.ButtonOut(100,4,204,26,mtext[2]);
Mnu.ButtonOut(214,4,302,26,mtext[3]); Mnu.ButtonOut(312,4,456,26,mtext[4]);
Mnu.ButtonOut(466,4,538,26,mtext[5]); Mnu.ButtonOut(14,445,86,465,'Start');
Mnu.ButtonOut(14,420,86,440,'Sprung'); Mnu.ButtonOut(14,395,86,415,'Zeichnen');
Mnu.ButtonIn(14,370,86,390,''); Mnu.ButtonOut(15,371,85,389,'Vollbild');
End;
{-----Prozeduren für Umschaltung zwischen Vollbildschirm und
  Bildschirm mit Titelleiste und Hauptfenster-----}
Procedure DrawScreen;
Begin
  SetViewPort(101,31,maxX,maxY,True);
End;
Procedure FullScreen;
Begin
  SetViewPort(0,0,maxX,maxY,True);
End;
{-----Variableninitialisierung (Variablen werden auf Standardwerte gesetzt)-----}
Procedure Var_Init;
Var i,j: Integer;
Begin
  maxX:=getmaxx; maxY:=getmaxy;
  ZellenX:=15;
  ZellenY:=15;
  ZellenZ:=15;
  ux:=270; uy:=200;
  Cvar[1,1].wert:=0; Cvar[1,2].wert:=1;
  Cvar[1,3].wert:=2; Cvar[1,4].wert:=3;
  DrawStop:=False; MausKlick:=True; ZellCheckZ:=True;
  ZellCheckY:=True; EndOfProg:=False; FullScreenChk:=False;
  RasterChk:=False; RahmenChk:=True; FeldTypChk:=True;
  For i:= 1 to 4 Do Begin
    Cvar[1,i].Exp1:=1;
    Cvar[1,i].Exp2:=1;
    Cvar[1,i].exp3[1]:=i-1;
    Cvar[1,i].CellSwrch:=True;
  End;
  For i:= 2 To 5 Do
    For j:= 1 To 4 Do Begin
      Cvar[i,j].exp1:=0;
      Cvar[i,j].exp2:=0;
      Cvar[i,j].exp3[1]:=0;
    End;
  End;

```

```

    Cvar[i,j].exp3[2]:=0;
    Cvar[i,j].CellSwrch:=False
end;
For i:= 1 To 5 Do
  For j:= 1 To 4 Do
    Cvar[i,j].prior:=j-1;
End;
{-----Initialisierung der Variablen für die Zellenachbarn-----}
Procedure NeighborInit;
Var i,j: Integer;
Begin
  seqanz:=1;
  For i:= 1 To 4 Do Begin
    nb[i,1]:=True;
    nb[i,2]:=True;
    nb[i,3]:=True;
    nb[i,4]:=False;
  End;
End;
{-----Zurücksetzen der Variablen des Konfigurationseditor-----}
Procedure ResetVar;
Var i,j: Integer;
Begin
  For i:= 1 to 5 Do
    For j:= 1 to 4 Do Begin
      Cvar[i,j].CellSwrch:=False;
      Cvar[i,j].exp1:=0;
      Cvar[i,j].exp2:=0;
      Cvar[i,j].exp3[1]:=0;
      Cvar[i,j].exp3[2]:=0;
      Cvar[i,j].prior:=j-1;
    End;
  End;
End;
{-----Funktion für die Ermittlung der größten Variablen-----}
Function GetBiggestCell(a,b,c:Byte):Byte;
Begin
  If c>=b Then b:=c;
  If b>a Then a:=b;
  GetBiggestCell:=a;
End;
{-----Variableninitialisierung für die Berechnung der dreidimensionalen Darstellung-----}
Procedure Var_Ausrechnen;
Begin
  ZellFaktor:=trunc(Zoom_Faktor/(getbiggestcell(ZellenX,ZellenY,ZellenZ)/2));
  DW:=pi*DrehWinkel/180; KW:=pi*KippWinkel/180;
  CosDW:=cos(DW); SinDW:=sin(DW);
  CosKW:=cos(KW); SinKW:=sin(KW);
  t[1]:=CosDW*CosKW; t[2]:=-SinDW*CosKW; t[3]:=-SinKW;
  t[4]:=SinDW; t[5]:=CosDW; t[6]:=0;
  t[7]:=CosDW*SinKW; t[8]:=-SinDW*SinKW; t[9]:=CosKW;
End;
{-----Umsetzung der dreidimensionalen Koordinaten auf Bildschirmkoordinaten-----}
Procedure GetXY(XKoor,YKoor,ZKoor:Integer;a,b,c:ShortInt;var x,y:Integer);

```

```

Var Xx,Yy,Zz,d : Real;
Begin
  XKoor:=-Zoom_Faktor+ZellFaktor*XKoor+a;
  YKoor:=-Zoom_Faktor+ZellFaktor*YKoor+b;
  ZKoor:=-Zoom_Faktor-ZellFaktor*ZKoor+c;
  xx:=t[1]*xkoor+t[2]*ykoor+t[3]*zkoor;
  yy:=t[4]*xkoor+t[5]*ykoor;
  zz:=t[7]*xkoor+t[8]*ykoor+t[9]*zkoor;
  d:=Distanz/(Distanz-xx);
  x:=ux+round(d*yy);
  y:=uy+round(d*zz);
End;
{----Kennzeichnen der aktuellen Ebene im dreidimensionalen Bereich des Zelleditors----}
Procedure DrawActLevel(typ,level:Byte);
Var x1,y1,x2,y2,x3,y3,x4,y4:Integer;
Begin
  SetColor(14);
  SetLineStyle(0,0,3);
  GetXY(ZellenX,0,0,4,0,0,x1,y1);
  GetXY(ZellenX,0,ZellenZ,4,0,-4,x2,y2); Line(x1,y1,x2,y2);
  GetXY(0,0,ZellenZ,0,0,-4,x1,y1);   Line(x2,y2,x1,y1);
  GetXY(0,ZellenY,ZellenZ,0,4,-4,x2,y2); Line(x1,y1,x2,y2);
  GetXY(0,ZellenY,0,0,4,0,x1,y1);   Line(x1,y1,x2,y2);
  GetXY(ZellenX,ZellenY,0,4,4,0,x2,y2); Line(x1,y1,x2,y2);
  GetXY(ZellenX,0,0,4,0,0,x1,y1);   Line(x1,y1,x2,y2);
  SetLineStyle(0,0,2);
  SetColor(15);
  Case Typ of
  3:Begin
    GetXY(level-1,0,ZellenZ,0,0,-4,x1,y1);
    GetXY(level,0,ZellenZ,0,0,-4,x2,y2);
    GetXY(level-1,ZellenY,0,0,4,0,x3,y3);
    GetXY(level,ZellenY,0,0,4,0,x4,y4);
    line(x1,y1,x2,y2);
    line(x3,y3,x4,y4);
  End;
  2:Begin
    GetXY(ZellenX,level-1,0,4,0,0,x1,y1);
    GetXY(ZellenX,level,0,4,0,0,x2,y2);
    GetXY(0,level-1,ZellenZ,0,0,-4,x3,y3);
    GetXY(0,level,ZellenZ,0,0,-4,x4,y4);
    Line(x1,y1,x2,y2);
    Line(x3,y3,x4,y4);
  End;
  1:Begin
    GetXY(ZellenX,0,level-1,4,0,0,x1,y1);
    GetXY(ZellenX,0,level,4,0,0,x2,y2);
    GetXY(0,ZellenY,level-1,0,4,0,x3,y3);
    GetXY(0,ZellenY,level,0,4,0,x4,y4);
    Line(x1,y1,x2,y2);
    Line(x3,y3,x4,y4);
  End;
End;

```

```

    SetLineStyle(0,0,1);
End;
{-----Zeichnen des dreidimensionalen Rasters-----}
Procedure Raster(Flag:Boolean);
Var x1,y1,x2,y2,x3,y3,x4,y4:Integer;
    i:Integer;
Begin
    If Flag=True Then Begin
        SetColor(13);
        For i:= 0 To ZellenX Do Begin
            GetXY(i,0,0,0,0,x1,y1);
            GetXY(i,0,ZellenZ,0,0,x2,y2);
            GetXY(i,ZellenY,0,0,0,x3,y3);
            Line(x1,y1,x2,y2);
            Line(x1,y1,x3,y3);
        End;
        For i:= 0 To ZellenY Do Begin
            GetXY(0,i,0,0,0,x1,y1);
            GetXY(0,i,ZellenZ,0,0,x2,y2);
            GetXY(ZellenX,i,0,0,0,x3,y3);
            Line(x1,y1,x2,y2);
            Line(x1,y1,x3,y3);
        End;
        For i:= 1 To ZellenZ Do Begin
            GetXY(0,0,i,0,0,x1,y1);
            GetXY(0,ZellenY,i,0,0,x2,y2);
            GetXY(ZellenX,0,i,0,0,x3,y3);
            Line(x1,y1,x2,y2);
            Line(x1,y1,x3,y3);
        End;
    End;
End;
Procedure RasterZeichnen(Flag:Boolean);
Begin
    If Flag Then Begin
        M.Show(False);
        DrawScreen;
        ClearViewport;
        Var_Ausrechnen;
        Raster(Flag);
        Fullscreen;
        M.Show(True);
    End;
End;
{-----Fenster für diverse Mitteilungen-----}
Procedure MessageWindow(title,txt:String;taste:Integer);
Var size: Word;
    p: Pointer;
    len,x1,x2: Integer;
Begin
    FullScreen;
    len:=TextWidth(txt);
    x1:=320-len div 2-20;

```

```

x2:=320+len div 2+20;
size:=ImageSize(x1,200,x2,254);
GetMem(p,size);
GetImage(x1,200,x2,254,p^);
Mnu.Win(x1,200,x2,254,15,14,13);
Mnu.Win(x1+4,204,x2-4,220,13,14,15);
Mnu.Win(x1+4,224,x2-4,250,13,14,15);
SetText(320-textwidth(title) div 2,208,title,6,0);
SetText(x1+20,233,txt,15,0);
If taste=9 Then Begin
  FreeMem(p,size);
  Exit;
End;
Repeat
  M.Status(mx,my,mt);
Until mt=taste;
Putimage(x1,200,p^,0);
Freemem(p,size);
End;
{----Fenster mit Ja- und Nein-Schaltflächen----}
Procedure QuestionWindow(title,txt:String;var antwort:Boolean);
var size: Word;
    p: Pointer;
    len,x1,x2: Integer;
    yes,no: Boolean;
Begin
  FullScreen;
  len:=textwidth(txt);
  x1:=320-len div 2-20;
  x2:=320+len div 2+20;
  size:=ImageSize(x1,200,x2,280);
  GetMem(p,size);
  GetImage(x1,200,x2,280,p^);
  Mnu.Win(x1,200,x2,280,15,14,13);
  Mnu.Win(x1+4,204,x2-4,220,13,14,15);
  Mnu.Win(x1+4,224,x2-4,276,13,14,15);
  Mnu.ButtonOut(270,250,310,270,'JA');
  Mnu.ButtonOut(330,250,370,270,'NEIN');
  SetText(320-textwidth(title) div 2,208,title,6,0);
  SetText(x1+20,233,txt,15,0);
  no:=False; yes:=False;
  M.SetHand;
  M.Show(True);
  Repeat
    M.Status(mx,my,mt);
    Mnu.GetButton(270,250,310,270,linketaste,mx,my,mt,'JA',yes);
    Mnu.GetButton(330,250,370,270,linketaste,mx,my,mt,'NEIN',no);
  Until Yes or No;
  M.Show(False);
  If Yes Then antwort:=True Else antwort:=False;
  Putimage(x1,200,p^,0);
  Freemem(p,size);
End;

```

```
{-----Feststellen des neuen Zustands der Zelle,  
gemäß den Regeln und den aktivierten Nachbarn-----}
```

```
Procedure Zelle_Lesen(x,y,z:Byte);
```

```
Var p: Array[1..27] Of Byte;
```

```
tb,tr,tg,ts,i,t,geswert: Integer;
```

```
alle,fl,kt,ek,cc: Byte;
```

```
xmin,ymin,zmin,xmax,ymax,zmax: Byte;
```

```
{-----Flächennachbarn einlesen-----}
```

```
Procedure GetFINb(x,y,z,seq:Byte);
```

```
Begin
```

```
  If nb[seq,1] Then Begin
```

```
    p[1]:=Feld[x,y,zmin];
```

```
    p[2]:=Feld[xmin,y,z];
```

```
    p[3]:=Feld[xmax,y,z];
```

```
    p[4]:=Feld[x,ymin,z];
```

```
    p[5]:=Feld[x,ymax,z];
```

```
    p[6]:=Feld[x,y,zmax];
```

```
  End;
```

```
End;
```

```
{-----Kantennachbarn einlesen-----}
```

```
Procedure GetKtNb(x,y,z,seq:Byte);
```

```
Begin
```

```
  If nb[seq,2] Then Begin
```

```
    p[1+fl]:=Feld[x,ymin,zmin];
```

```
    p[2+fl]:=Feld[xmin,y,zmin];
```

```
    p[3+fl]:=Feld[xmax,y,zmin];
```

```
    p[4+fl]:=Feld[x,ymax,zmin];
```

```
    p[5+fl]:=Feld[xmin,ymin,z];
```

```
    p[6+fl]:=Feld[xmax,ymin,z];
```

```
    p[7+fl]:=Feld[xmin,ymax,z];
```

```
    p[8+fl]:=Feld[xmax,ymax,z];
```

```
    p[9+fl]:=Feld[x,ymin,zmax];
```

```
    p[10+fl]:=Feld[xmin,y,zmax];
```

```
    p[11+fl]:=Feld[xmax,y,zmax];
```

```
    p[12+fl]:=Feld[x,ymax,zmax];
```

```
  End;
```

```
End;
```

```
{-----Ecknachbarn einlesen-----}
```

```
Procedure GetEkNb(seq:Byte);
```

```
Begin
```

```
  If nb[seq,3] Then Begin
```

```
    p[1+fl+kt]:=Feld[xmin,ymin,zmin];
```

```
    p[2+fl+kt]:=Feld[xmin,ymax,zmin];
```

```
    p[3+fl+kt]:=Feld[xmax,ymin,zmin];
```

```
    p[4+fl+kt]:=Feld[xmax,ymax,zmin];
```

```
    p[5+fl+kt]:=Feld[xmin,ymin,zmax];
```

```
    p[6+fl+kt]:=Feld[xmin,ymax,zmax];
```

```
    p[7+fl+kt]:=Feld[xmax,ymin,zmax];
```

```
    p[8+fl+kt]:=Feld[xmax,ymax,zmax];
```

```
  End;
```

```
End;
```

```
{-----Betrachtete Zelle einlesen-----}
```

```

Procedure GetCentral(x,y,z,seq:Byte);
Begin
  If nb[seq,4] Then p[1+fl+kt+ek]:=Feld[x,y,z];
End;
{----Nachbarn einlesen-----}
Procedure GetCells(x,y,z,seq:Byte);
Begin
  If nb[seq,1] Then fl:=6 Else fl:=0;
  If nb[seq,2] Then kt:=12 Else kt:=0;
  If nb[seq,3] Then ek:=8 Else ek:=0;
  If nb[seq,4] Then cc:=1 Else cc:=0;
  If FeldTypChk Then Begin
    xmin:=x-1; xmax:=x+1;
    ymin:=y-1; ymax:=y+1;
    zmin:=z-1; zmax:=z+1;
  End Else Begin
    If x=1 Then xmin:=ZellenX Else xmin:=x-1;
    If x=ZellenX Then xmax:=1 Else xmax:=x+1;
    If y=1 Then ymin:=ZellenY Else ymin:=y-1;
    If y=ZellenY Then ymax:=1 Else ymax:=y+1;
    If z=1 Then zmin:=ZellenZ Else zmin:=z-1;
    If z=ZellenZ Then zmax:=1 Else zmax:=z+1;
  End;
  GetFINb(x,y,z,seq);
  GetKtNb(x,y,z,seq);
  GetEkNb(seq);
  GetCentral(x,y,z,seq);
End;
Function Exp2Typ(typ,cell,exp1,a,b:Byte):Boolean;
Var wert:Byte;

Function ZiffernSumme(zahl:Byte):Byte;
Var st: String[2];
    s1,s2: Byte;
    Code: Integer;
Begin
  str(zahl:2,st);
  val(copy(st,1,1),s1,code);
  val(copy(st,2,1),s2,code);
  ZiffernSumme:=s1+s2;
End;

Begin
  If exp1=1 Then wert:=t
  Else If exp1=2 Then wert:=tb
  Else If exp1=3 Then wert:=tr
  Else If exp1=4 Then wert:=tg
  Else If exp1=5 Then wert:=ts
  Else If exp1=6 Then wert:=geswert;
  If Cvar[typ,cell].exp2=1 Then If wert=a Then exp2Typ:=True
    Else exp2Typ:=False Else
  If Cvar[typ,cell].exp2=2 Then If wert<>a Then exp2Typ:=True
    Else exp2Typ:=False Else

```

```

If Cvar[typ,cell].exp2=3 Then If wert<a Then exp2typ:=True
    Else exp2typ:=False Else
If Cvar[typ,cell].exp2=4 Then If wert>a Then exp2typ:=True
    Else exp2typ:=False Else
If Cvar[typ,cell].exp2=5 Then If ((wert>a) And (wert<b)) Then exp2typ:=True
    Else exp2typ:=False Else
If Cvar[typ,cell].exp2=6 Then If ((wert<a) Or (wert>b)) Then exp2typ:=True
    Else exp2typ:=False Else
If Cvar[typ,cell].exp2=7 Then If wert mod a=b Then exp2typ:=True
    Else exp2typ:=False Else
If Cvar[typ,cell].exp2=8 Then If ZiffernSumme(wert)<a Then exp2typ:=True Else exp2typ:=False;
End;
{----Hauptteil der Prozedur Zelle_Lesen----}
Begin
tb:=0; tr:=0; tg:=0; ts:=0; geswert:=0; t:=0;
GetCells(x,y,z,(Datum mod SeqAnz)+1);
alle:=fl+kt+ek+cc;
For i:=1 to alle Do Begin
If p[i]=1 Then Begin
    Inc(tb);
    Inc(geswert,Cvar[1,2].wert);
End;
If p[i]=2 Then Begin
    Inc(tr);
    Inc(geswert,Cvar[1,3].wert);
End;
If p[i]=3 Then Begin
    Inc(tg);
    Inc(geswert,Cvar[1,4].wert);
End;
If p[i]=0 Then Begin
    Inc(ts);
    Inc(geswert,Cvar[1,1].wert);
End;
End;
t:=tb+tr+tg;
If Cvar[1,4].CellSwch Then
If Exp2Typ(1,4,Cvar[1,4].exp1,Cvar[1,4].exp3[1],Cvar[1,4].exp3[2]) Then
    NeuFeld[x,y,z]:=Cvar[1,4].prior;
If Cvar[1,3].CellSwch Then
If Exp2Typ(1,3,Cvar[1,3].exp1,Cvar[1,3].exp3[1],Cvar[1,3].exp3[2]) Then
    NeuFeld[x,y,z]:=Cvar[1,3].prior;
If Cvar[1,2].CellSwch Then
If Exp2Typ(1,2,Cvar[1,2].exp1,Cvar[1,2].exp3[1],Cvar[1,2].exp3[2]) Then
    NeuFeld[x,y,z]:=Cvar[1,2].prior;
If Cvar[1,1].CellSwch Then
If Exp2Typ(1,1,Cvar[1,1].exp1,Cvar[1,1].exp3[1],Cvar[1,1].exp3[2]) Then
    NeuFeld[x,y,z]:=Cvar[1,1].prior;
If Feld[x,y,z]=0 Then Begin
If Cvar[5,4].CellSwch Then
If Exp2Typ(5,4,Cvar[5,4].exp1,Cvar[5,4].exp3[1],Cvar[5,4].exp3[2]) Then
    NeuFeld[x,y,z]:=Cvar[5,4].prior;
If Cvar[5,3].CellSwch Then

```



```

    If Exp2Typ(5,3,Cvar[5,3].exp1,Cvar[5,3].exp3[1],Cvar[5,3].exp3[2]) Then
      NeuFeld[x,y,z]:=Cvar[5,3].prior;
  If Cvar[5,2].CellSwch Then
    If Exp2Typ(5,2,Cvar[5,2].exp1,Cvar[5,2].exp3[1],Cvar[5,2].exp3[2]) Then
      NeuFeld[x,y,z]:=Cvar[5,2].prior;
  If Cvar[5,1].CellSwch Then
    If Exp2Typ(5,1,Cvar[5,1].exp1,Cvar[5,1].exp3[1],Cvar[5,1].exp3[2]) Then
      NeuFeld[x,y,z]:=Cvar[5,1].prior;
End;
If Feld[x,y,z]=1 Then Begin
  If Cvar[2,4].CellSwch Then
    If Exp2Typ(2,4,Cvar[2,4].exp1,Cvar[2,4].exp3[1],Cvar[2,4].exp3[2]) Then
      NeuFeld[x,y,z]:=Cvar[2,4].prior;
  If Cvar[2,3].CellSwch Then
    If Exp2Typ(2,3,Cvar[2,3].exp1,Cvar[2,3].exp3[1],Cvar[2,3].exp3[2]) Then
      NeuFeld[x,y,z]:=Cvar[2,3].prior;
  If Cvar[2,2].CellSwch Then
    If Exp2Typ(2,2,Cvar[2,2].exp1,Cvar[2,2].exp3[1],Cvar[2,2].exp3[2]) Then
      NeuFeld[x,y,z]:=Cvar[2,2].prior;
  If Cvar[2,1].CellSwch Then
    If Exp2Typ(2,1,Cvar[2,1].exp1,Cvar[2,1].exp3[1],Cvar[2,1].exp3[2]) Then
      NeuFeld[x,y,z]:=Cvar[2,1].prior;
End;
If Feld[x,y,z]=2 Then Begin
  If Cvar[3,4].CellSwch Then
    If Exp2Typ(3,4,Cvar[3,4].exp1,Cvar[3,4].exp3[1],Cvar[3,4].exp3[2]) Then
      NeuFeld[x,y,z]:=Cvar[3,4].prior;
  If Cvar[3,3].CellSwch Then
    If Exp2Typ(3,3,Cvar[3,3].exp1,Cvar[3,3].exp3[1],Cvar[3,3].exp3[2]) Then
      NeuFeld[x,y,z]:=Cvar[3,3].prior;
  If Cvar[3,2].CellSwch Then
    If Exp2Typ(3,2,Cvar[3,2].exp1,Cvar[3,2].exp3[1],Cvar[3,2].exp3[2]) Then
      NeuFeld[x,y,z]:=Cvar[3,2].prior;
  If Cvar[3,1].CellSwch Then
    If Exp2Typ(3,1,Cvar[3,1].exp1,Cvar[3,1].exp3[1],Cvar[3,1].exp3[2]) Then
      NeuFeld[x,y,z]:=Cvar[3,1].prior;
End;
If Feld[x,y,z]=3 Then Begin
  If Cvar[4,4].CellSwch Then
    If Exp2Typ(4,4,Cvar[4,4].exp1,Cvar[4,4].exp3[1],Cvar[4,4].exp3[2]) Then
      NeuFeld[x,y,z]:=Cvar[4,4].prior;
  If Cvar[4,3].CellSwch Then
    If Exp2Typ(4,3,Cvar[4,3].exp1,Cvar[4,3].exp3[1],Cvar[4,3].exp3[2]) Then
      NeuFeld[x,y,z]:=Cvar[4,3].prior;
  If Cvar[4,2].CellSwch Then
    If Exp2Typ(4,2,Cvar[4,2].exp1,Cvar[4,2].exp3[1],Cvar[4,2].exp3[2]) Then
      NeuFeld[x,y,z]:=Cvar[4,2].prior;
  If Cvar[4,1].CellSwch Then
    If Exp2Typ(4,1,Cvar[4,1].exp1,Cvar[4,1].exp3[1],Cvar[4,1].exp3[2]) Then
      NeuFeld[x,y,z]:=Cvar[4,1].prior;
End;
End;
{-----obige Prozedur für alle Zellen des Raumes wiederholen-----}

```

```

Procedure ZellenLesen;
Var i,j,k:Byte;
Begin
  For i:= 1 To ZellenZ Do
    For j:= 1 To ZellenY Do
      For k:= 1 To ZellenX Do
        Zelle_lesen(k,j,i);
      End;
    End;
  End;
  {-----Festsetzen der Rahmenfarbe und der Flächenfarbe der Zelle-----}
  Procedure Color(c1,c2:Integer);
  Begin
    SetFillStyle(1,c1);
    If RahmenChk Then SetColor(c2) Else SetColor(c1);
  End;
  {-----Zeichnen der dreidimensionalen Zelle auf dem Bildschirm-----}
  Procedure Zelle_Setzen(x,y,z:Integer);
  Var p : Array[1..3,1..5] Of Pointtype;
      ep: Array[1..1] Of Pointtype;

  Function InMaxX(p1,p2,p3,p4:Integer):Boolean;
  Begin
    If (p1<maxX) Or (p2<maxX) Or (p3<maxX) Or (p4<maxX) Then InMaxX:=True
      Else InMaxX:=False;
  End;

  Begin
    GetXY(x,y-1,z,-sp,sp,sp,p[1,2].x,p[1,2].y);
    GetXY(x-1,y,z,sp,-sp,sp,p[1,4].x,p[1,4].y);
    GetXY(x-1,y-1,z,sp,sp,sp,p[1,1].x,p[1,1].y);
    ep[1]:=p[1,1];
    GetXY(x,y,z,-sp,-sp,sp,p[1,3].x,p[1,3].y);
    p[1,5]:=p[1,1];
    p[2,1]:=p[1,2];
    p[2,2]:=p[1,3];
    GetXY(x,y,z-1,-sp,-sp,-sp,p[2,3].x,p[2,3].y);
    GetXY(x,y-1,z-1,-sp,sp,-sp,p[2,4].x,p[2,4].y);
    p[2,5]:=p[2,1];
    p[3,1]:=p[2,3];
    GetXY(x-1,y,z-1,sp,-sp,-sp,p[3,2].x,p[3,2].y);
    p[3,3]:=p[1,4];
    p[3,4]:=p[1,3];
    p[3,5]:=p[3,1];
    If p[1,3].y<p[1,1].y Then Begin
      GetXY(x-1,y-1,z-1,sp,sp,-sp,p[1,1].x,p[1,1].y);
      p[1,2]:=p[2,4];
      p[1,3]:=p[2,3];
      p[1,4]:=p[3,2];
      p[1,5]:=p[1,1];
      ZellCheckZ:=False;
    End;
    If InMaxX(p[3,4].x,p[3,2].x,p[3,1].x,p[3,3].x) Then
      If ((p[3,4].x>p[3,2].x) and (ZellCheckZ)) Or
        ((p[3,1].x>p[3,3].x) and (ZellCheckZ=False)) Then Begin

```

```

    getxy(x-1,y-1,z-1,sp,sp,-sp,p[3,2].x,p[3,2].y);
    p[3,1]:=p[2,4];
    p[3,3]:=ep[1];
    p[3,4]:=p[2,1];
    p[3,5]:=p[3,1];
    ZellCheckY:=False;
End;
If Feld[x,y,z]=1 Then Begin
    color(3,10); fillpoly(5,p[3]);
    color(2,10); fillpoly(5,p[2]);
    color(1,10); fillpoly(5,p[1]);
End Else
    If Feld[x,y,z]=2 Then Begin
        color(6,11); fillpoly(5,p[3]);
        color(5,11); fillpoly(5,p[2]);
        color(4,11); fillpoly(5,p[1]);
    End Else
        If Feld[x,y,z]=3 Then Begin
            color(9,12); fillpoly(5,p[3]);
            color(8,12); fillpoly(5,p[2]);
            color(7,12); fillpoly(5,p[1]);
        End;
End;
{-----obige Prozedur für alle Zellen wiederholen-----}
Procedure ZellenSetzen(flag:Boolean);

{-----Prozeduren für Änderung des "Warten"- und des "Abbruch"-Status-----}
Procedure MausAbfrage;
Begin
    If Flag Then Begin
        M.Status(mx,my,mt);
        If mt=rechtetaste Then Begin
            DrawStop:=not DrawStop;
            If DrawStop Then MessageWindow('ABBRUCH','AKTIVIERT',keinetaste)
                Else MessageWindow('ABBRUCH','DEAKTIVIERT',keinetaste);
            If FullScreenChk=False Then drawscreen;
        End;
        If (mt=linketaste) and (DrawStop=False) Then Begin
            mausklick:=not mausklick;
            If mausklick Then MessageWindow('WARTEN','AKTIVIERT',keinetaste)
                Else MessageWindow('WARTEN','DEAKTIVIERT',keinetaste);
            If FullScreenChk=False Then drawscreen;
        End;
    End;
End;
End;

Var z,j,k,y: Byte;
Begin
    z:=0;
    Repeat
        y:=0;
        ZellCheckY:=True;
        Inc(z);

```

```

Repeat
  Inc(y);
  For k:= 1 to ZellenX Do Begin
    If Feld[k,y,z]<>0 Then zelle_setzen(k,y,z);
    MausAbfrage;
  End;
Until (ZellCheckY=False) Or (y=ZellenY);
If ZellCheckY=False Then Begin
  For j:= ZellenY Downto y Do
    For k:= 1 to ZellenX Do Begin
      If Feld[k,j,z]<>0 Then zelle_setzen(k,j,z);
      MausAbfrage;
    End;
  End;
End;
Until (ZellCheckZ=False) Or (z=ZellenZ);
If (ZellCheckZ=False) Then
  For i:=ZellenZ Downto z Do Begin
    y:=0;
    ZellCheckY:=True;
    Repeat
      Inc(y);
      For k:= 1 to ZellenX Do Begin
        If Feld[k,y,i]<>0 Then zelle_setzen(k,y,i);
        MausAbfrage;
      End;
    Until (ZellCheckY=False) Or (y=ZellenY);
    If ZellCheckY=False Then Begin
      For j:= ZellenY Downto y Do
        For k:= 1 to ZellenX Do Begin
          If Feld[k,j,i]<>0 Then zelle_setzen(k,j,i);
          MausAbfrage;
        End;
      End;
    End;
  End;
ZellCheckZ:=True;
ZellCheckY:=True;
End;
{----Zurücksetzen der Zustände der Zellen des gesamten Raumes----}
Procedure FeldReset;
Var i,j,k: Integer;
Begin
  For i:= 0 To MaxZellAnzahl_Z+1 Do
    For j:= 0 To MaxZellAnzahl_Y+1 Do
      For k:= 0 To MaxZellAnzahl_X+1 Do Begin
        NeuFeld[k,j,i]:=0;
        Feld[i,j,k]:=0;
      End;
    End;
  End;
End;
{----Darstellung der Anzahl der lebenden Zellen im Statusfenster----}
Procedure WriteCellStatus(flag:Boolean);
var st: String;
    i,j,k,zgr,zbl,zrt: Integer;
Begin

```

```

zbl:=0; zrt:=0; zgr:=0;
For i:= 1 To ZellenZ Do
  For j:= 1 To ZellenY Do
    For k:= 1 To ZellenX Do Begin
      If Feld[k,j,i]=1 Then Inc(zbl) Else
        If Feld[k,j,i]=2 Then Inc(zrt) Else
          If Feld[k,j,i]=3 Then Inc(zgr);
    End;
  If Flag=False Then Begin
    color(14,14); solidbar(51,65,93,110);
    str(Datum:5,st); SetColor(0); ShowTextXY(54,65,st);
    str(zbl:5,st); SetColor(1); ShowTextXY(54,78,st);
    str(zrt:5,st); SetColor(4); ShowTextXY(54,91,st);
    str(zgr:5,st); SetColor(7); ShowTextXY(54,104,st);
  End;
End;
{-----Speichern eines Bildschirmausschnitts-----}
Procedure GetScreen;
Begin
  size1:=ImageSize(0,0,maxX,30);
  GetMem(p1,size1);
  GetImage(0,0,maxX,30,p1^);
  size2:=ImageSize(0,31,100,maxY);
  GetMem(p2,size2);
  GetImage(0,31,100,maxY,p2^);
End;
{-----Setzen des gespeicherten Bildschirmausschnitts-----}
Procedure PutScreen;
Begin
  PutImage(0,0,p1^,0); PutImage(0,31,p2^,0);
  FreeMem(p1,size1); FreeMem(p2,size2);
End;
{-----Warten auf Mausklick nach Beendigung des graphischen Aufbaus-----}
Procedure WaitForMouseClicked;
var size: Word;
    p: Pointer;
Begin
  FullScreen;
  size:=ImageSize(589,439,639,479);
  GetMem(p,size);
  GetImage(589,439,639,479,p^);
  Mnu.Win(589,439,639,479,15,14,13);
  Mnu.ButtonOut(594,444,612,464,'L');
  Mnu.ButtonOut(616,444,634,464,'R');
  SetText(594,468,'GO AB',0,15);
  Repeat
    M.Status(mx,my,mt);
    If mt=rechtetaste Then DrawStop:=True;
  Until (mt=linketaste) Or (mt=rechtetaste);
  If mt=linketaste Then Mnu.ButtonIn(594,444,612,464,'L');
  If mt=rechtetaste Then Mnu.ButtonIn(616,444,634,464,'R');
  Repeat
    M.Status(mx,my,mt);

```

```

Until mt=keinetaste;
PutImage(589,439,p^,0);
FreeMem(p,size);
If FullScreenChk=False Then DrawScreen;
End;
{-----Zeichnen der Zellen, gemäß dem "Vollbild"-Status-----}
Procedure Draw_Cells;
Begin
If FullScreenChk=True Then Begin
fullscreen;
ux:=320;
uy:=215;
Zoom_Faktor:=Zoom_Faktor+25;
var_ausrechnen;
getscreen;
End Else drawscreen;
clearviewport;
raster(RasterChk);
zellensetzen(False);
If FullScreenChk=True Then Begin
WaitForMouseClicked;
putscreen;
WriteCellStatus(False);
ux:=270;
uy:=200;
Zoom_Faktor:=Zoom_Faktor-25;
Var_Ausrechnen;
End;
FullScreen;
End;
{-----Start der Simulation-----}
Procedure StartDraw_Cells;
Begin
M.Show(False);
DrawStop:=False;
If FullScreenChk Then Begin
ux:=320;
uy:=215;
Zoom_Faktor:=Zoom_Faktor+25;
Var_Ausrechnen;
FullScreen;
GetScreen;
End Else DrawScreen;
Repeat
ZellenLesen;
Feld:=NeuFeld;
Inc(Datum);
FullScreen;
WriteCellStatus(FullScreenChk);
If FullScreenChk=False Then DrawScreen;
ClearViewPort;
Raster(RasterChk);
ZellenSetzen(True);

```

```

    If MausKlick and (DrawStop=False) Then WaitForMouseClicked;
Until DrawStop;
If FullScreenChk Then Begin
    PutScreen;
    WriteCellStatus(False);
    ux:=270;
    uy:=200;
    Zoom_Faktor:=Zoom_Faktor-25;
    Var_Ausrechnen;
End;
FullScreen;
M.Show(True);
End;
{----Start der Simulation ohne graphische Darstellung nach jeder Generation----}
Procedure Sprung;
Begin
    M.Show(False);
    MessageWindow('SIMULATION LÄUFT','<ESC> DRÜCKEN UM ZU BEENDEN',9);
    Repeat
        ZellenLesen;
        Feld:=NeuFeld;
        Inc(Datum);
        WriteCellStatus(False);
    Until keypressed and (readkey=#27);
    If FullScreenChk Then Begin
        ux:=320;
        uy:=215;
        Zoom_Faktor:=Zoom_Faktor+25;
        Var_Ausrechnen;
        FullScreen;
        GetScreen;
    End Else DrawScreen;
    ClearViewPort;
    Raster(RasterChk);
    ZellenSetzen(False);
    If FullScreenChk=True Then Begin
        WaitForMouseClicked;
        PutScreen;
        ux:=270;
        uy:=200;
        Zoom_Faktor:=Zoom_Faktor-25;
        Var_Ausrechnen;
    End;
    FullScreen;
    M.Show(True);
End;
{----Fenster für Ein- und Ausgabe von Dateien----}
Procedure LadenSichernWindow(typ,bob:Boolean;var Code:Boolean);
const x1=150;
      y1=95;
      x2=490;
      y2=381;
      fmax=240;

```

```

var size: Word;
    p: Pointer;
    files: Array[1..fmax+1] Of String[12];
    path: String;
    zv,filestart,activef,oldf: Byte;
    x,y,oldx,oldy: Integer;

Procedure GetScreen;
Begin
    size:=ImageSize(x1,y1,x2,y2);
    GetMem(p,size);
    GetImage(x1,y1,x2,y2,p^);
End;
Procedure PutScreen;
Begin
    PutImage(x1,y1,p^,0);
    FreeMem(p,size);
End;
{-----Zeichen des Laden-/Sichern-Dialogfensters-----}
Procedure ScreenInit;
var txt: String;
Begin
    Mnu.Win(x1,y1,x2,y2,15,14,13);
    Mnu.Win(x1+4,y1+4,x2-4,y1+19,13,14,15);
    M.MWindow(x1,y1,x2,y2);
    SetText(160,131,'Name:',15,0);
    SetText(160,160,'Pfad:',15,0);
    SetText(160,189,'Dateien:',15,0);
    Mnu.Win(210,125,324,145,15,14,13);
    Mnu.Win(210,154,480,174,15,14,13);
    Mnu.ButtonOut(160,351,180,371,#17);
    Mnu.ButtonOut(370,351,390,371,#16);
    Mnu.ButtonOut(400,351,480,371,'ABBRUCH');
    If Bob Then txt:='Bild' Else txt:='Konfiguration';
    If Typ Then txt:=txt+' laden' Else txt:=txt+' speichern';
    SetText(x1+((x2-x1)-textwidth(txt)) div 2,y1+8,txt,15,0);
    If typ Then Mnu.ButtonOut(400,321,480,341,'LADEN')
        Else Mnu.ButtonOut(400,321,480,341,'SPEICHERN');
End;
{-----Initialisierung der String-Variablen-----}
Procedure ResetFiles;
Var i: Integer;
Begin
    For i:= 1 To fmax Do
        files[i]:= "";
    End;
End;
Function BackSlash(wort:String): String;
Begin
    If copy(wort,length(wort),1)<>'\' Then BackSlash:=wort+'\'
        Else BackSlash:=wort;
End;
{-----Lesen der Dateien im aktuellen Suchpfad-----}
Procedure GetFiles(path:String);

```



```

Var search: searchrec;
  i: Integer;
  erw: String[3];
Begin
  If bob Then erw:=erwpic Else erw:=erwbed;
  ResetFiles;
  zv:=0;
  FindFirst(BackSlash(path)+'*.*',anyfile,search);
  If DosError=0 Then Begin
    while DosError=0 do Begin
      If (copy(search.name,length(search.name)-2,3)=erw) or (search.attr=$10)
      Then Begin
        If search.name='.' Then FindNext(search);
        Inc(zv);
        If zv>=fmax Then break;
        FindNext(search);
      End Else Findnext(search);
    End;
  FindFirst(backslash(path)+'*.*',anyfile,search);
  i:=0;
  Repeat
    If (copy(search.name,length(search.name)-2,3)=erw) or (search.attr=$10)
    Then Begin
      Inc(i);
      If search.name='.' Then Findnext(search);
      If search.attr=directory Then files[i]:=search.name+'\
        Else files[i]:=search.name;
      FindNext(search);
    End Else FindNext(search)
  Until i=zv;
  End Else Begin
    zv:=1;
    files[1]:='<Keine>';
  End;
End;
{-----Sortieren der eingelesenen Dateien-----}
Procedure Sortieren;
var i,j,s: Integer;
  zwistr: String[12];
Begin
  for i:=2 To zv Do
    for j:=zv Downto i Do
      If files[j] < files[j-1] Then Begin
        zwistr:=files[j];
        files[j]:=files[j-1];
        files[j-1]:=zwistr;
      End;
    End;
  End;
Function GetCountNumber:String;
var i,c:Integer;
  erw:String;

Function LeadingZero(w : Word) : String;

```

```

var s:String;
Begin
  Str(w:0,s);
  Case Length(s) of
    1:s:='000'+s;
    2:s:='00'+s;
    3:s:='0'+s;
  End;
  LeadingZero:=s;
End;

Begin
  If bob Then erw:=erwpic Else erw:=erwbed;
  c:=1;
  For i:= 1 to zv do
    If files[i]='LIFE'+LeadingZero(c)+'.'+erw Then Inc(c);
    GetCountNumber:=LeadingZero(c);
  End;
  {-----Prozedur für die Leiste unter dem DateienFeld-----}
  Procedure Leiste(start,max:Byte);
  var x: Integer;
  Begin
    Mnu.Win(182,351,368,371,13,14,15);
    If max mod 24<>0 Then max:=(max-max mod 24)+1
      Else max:=max-23;
    x:=184+trunc(182/max*filestart*24);
    If x>350 Then x:=350;
    Mnu.Win(x,353,x+16,369,15,14,13);
  End;
  {-----Schreiben des Dateinamens im Feld "Datei"-----}
  Procedure WriteFileName(fnr:Byte);
  Begin
    SetColor(14);
    SetFillStyle(1,14);
    SolidBar(220,131,316,139);
    SetColor(0);
    ShowTextXY(220,131,files[fnr]);
  End;
  Procedure SetFileName(fn:String);
  Begin
    SetColor(14);
    SetFillStyle(1,14);
    SolidBar(220,131,316,139);
    SetColor(0);
    ShowTextXY(220,131,fn);
  End;
  {-----Schreiben der Dateinamen im Feld "Dateien"-----}
  Procedure WriteFiles(von,bis:Byte);
  var i,x,y: Integer;
  Begin
    M.Show(False);
    Mnu.Win(160,206,390,350,15,14,13);
    SetColor(0);

```

```

x:=0;
y:=0;
If bis>zv Then bis:=zv;
For i:= von To bis Do Begin
  If y=12 Then Begin
    x:=110;
    y:=0;
  End;
  If i=activef Then SetText(170+x,214+y*11,files[i],15,0)
  Else Begin
    SetColor(0);
    ShowTextXY(170+x,214+y*11,files[i]);
  End;
  Inc(y);
End;
Leiste(filestart,zv);
M.Show(True);
End;
{-----Funktion zur Überprüfung, ob Datei bereits existiert-----}
Function FileExists(FileName: String): Boolean;
Var f: File;
Begin
  {$I-}
  Assign(f, FileName);
  Reset(f); Close(f);
  {$I+}
  FileExists:= (IOResult = 0) and (FileName <> "");
End;
{-----Laden einer Datei-----}
Procedure LoadFile(filename:String;bob:Boolean;var errorcode:Boolean);
Var f1: File Of Byte;
    f2: File Of ConfigTyp;
    x,y,z: Integer;
    head: Byte;
Begin
  M.Show(False);
  If FileExists(filename) Then
    If Bob Then Begin
      Assign(f1,filename);
      Reset(f1);
      Read(F1,Head);
      If Head=Header1 Then Begin
        MessageWindow('Laden...','BITTE WARTEN',9);
        FeldReset;
        Datum:=0;
        Read(F1,ZellenX);
        Read(F1,ZellenY);
        Read(F1,ZellenZ);
        For x:= 1 to ZellenX do
          For y:= 1 to ZellenY do
            For z:= 1 to ZellenZ do
              Read(F1,Feld[x,y,z]);
            errorcode:=False;

```

```

    Var_Ausrechnen;
End Else Begin
    errorcode:=True;
    MessageWindow('FEHLER','FALSCHES DATEIFORMAT',linketaste);
    M.Setxy(320,240);
End;
Close(F1);
End Else Begin
    Assign(f2,filename);
    Reset(f2);
    MessageWindow('Laden...','BITTE WARTEN',9);
    For x:= 1 to 5 do
        For y:= 1 to 4 do
            Read(F2,Cvar[x,y]);
        errorcode:=False;
        Close(F2);
    End
Else Begin
    errorcode:=True;
    MessageWindow('FEHLER','DATEI NICHT GEFUNDEN',linketaste);
    M.SetXY(320,240);
End;
M.Show(True);
End;
{-----Funktion zur Überprüfung, ob Datei erstellt werden kann-----}
Function CreatingFile(FileName: String): Boolean;
Var F: File;
Begin
    {$I-}
    Assign(F,FileName);
    ReWrite(F); Close(F);
    {$I+}
    CreatingFile:= (IOResult=0) And (FileName<>"");
End;
{-----Speichern einer Datei-----}
Procedure SaveFile(filename:String;bob:Boolean;var errorcode:Boolean);
Var F1: File Of Byte;
    F2: File Of ConfigTyp;
    x,y,z: Integer;
    Exist,Quest: Boolean;
Begin
    Exist:=False; Quest:=True;
    For i:= 1 To zv Do
        If FileName=Files[i] Then Begin
            Exist:=True;
            Break;
        End;
    If (Exist And FileExists(filename)) Then Begin
        M.SetXY(320,240);
        M.Show(False);
        QuestionWindow('WARNUNG','DATEI EXISTIERT. ÜBERSCHREIBEN?',quest);
        M.Show(False);
    End;
End;

```

```

If Quest Then
  If CreatingFile(FileName) Then
    If Bob Then Begin
      MessageWindow('Speichern...','BITTE WARTEN',9);
      Assign(f1,filename);
      ReWrite(f1);
      Write(f1,header1);
      Write(f1,ZellenX);
      Write(f1,ZellenY);
      Write(f1,ZellenZ);
      For x:= 1 to ZellenX do
        For y:= 1 to ZellenY do
          For z:= 1 to ZellenZ do
            Write(f1,Feld[x,y,z]);
          Close(f1);
          ErrorCode:=False;
        End Else Begin
          MessageWindow('Speichern...','BITTE WARTEN',9);
          Assign(f2,filename);
          ReWrite(f2);
          For x:= 1 to 5 do
            For y:= 1 to 4 do
              Write(f2,Cvar[x,y]);
            Close(f2);
            ErrorCode:=False;
          End
        Else
          If Exist=False Then Begin
            ErrorCode:=True;
            MessageWindow('FEHLER','DATEI KANN NICHT ERSTELLT WERDEN',linketaste);
            M.SetXY(320,240);
          End;
        End;
      End;
    {-----Prozedur für Eingabe des Dateinamens per Tastatur-----}
    Procedure GetName(var filename:String;Var Code,Ok:Boolean);
    var taste: Char;
        x,i: Integer;
        oldname: String;
        erw: String[3];
        von: Byte;
    Begin
      If bob Then erw:=erwpic Else erw:=erwbed;
      M.Show(False);
      oldname:=files[activef];
      SetColor(0);
      OutTextXY(284,131,''+erw);
      x:=220;
      MoveTo(x,131);
      Repeat
        SetColor(0); Line(x,139,x+8,139);
        taste:=upcase(ReadKey);
        SetColor(14); Line(x,139,x+8,139);
        If (taste=#8) AND (x>=228) Then Begin

```

```

SetFillStyle(1,14);
SolidBar(getx-8,gety,getx,gety+8);
Delete(filename,length(filename),1);
Dec(x,8);
MoveTo(x,131);
End;
If not(taste in [#8,#13,#0]) And (x<=278) Then Begin
SetColor(0);
filename:=filename+taste;
OutText(taste);
Inc(x,8);
End;
Until taste=#13;
filename:=filename+'.'+erw;
If Not Typ Then Begin
SaveFile(Filename,bob,Code);
If Code=True Then filename:=oldname
Else Ok:=True;
End;
If Typ Then Begin
LoadFile(Filename,bob,Code);
If Code=True Then FileName:=OldName
Else Ok:=True;
End;
M.Show(False);
End;
{----Dateinamen eingeben----}
Procedure SearchFile(var code,ok:Boolean);
var chk: Boolean;
filename: String;
i: Integer;
von: Byte;
Begin
M.Show(False);
chk:=False;
Filename:="";
Mnu.Win(210,125,324,145,13,14,15);
GetName(Filename,code,ok);
Mnu.Win(210,125,324,145,15,14,13);
SetColor(0);
ShowTextXY(220,131,filename);
M.Show(True);
End;
{----Prozedur für Eingabe des Suchpfades per Tastatur----}
Procedure GetPath(var path:String);
Var st:String;
taste:char;
x:Integer;
Begin
Repeat
SetColor(14);
SetFillStyle(1,14);
SolidBar(220,160,468,168);

```

```

SetColor(0);
If length(path)<32 Then Begin
  ShowTextXY(220,160,path);
  x:=220+TextWidth(path);
End Else Begin
  ShowTextXY(220,160,copy(path,length(path)-30,31));
  x:=220+TextWidth(copy(path,length(path)-30,31));
End;
MoveTo(x,160);
Repeat
  SetColor(0);
  Line(x,168,x+8,168);
  taste:=upcase(ReadKey);
  SetColor(14);
  Line(x,168,x+8,168);
  If (taste=#8) AND (length(path)>=1) Then Begin
    Delete(path,length(path),1);
    If x>=228 Then Begin
      SetFillStyle(1,14);
      SolidBar(getx-8,gety,getx,gety+8);
      Dec(x,8);
      MoveTo(x,160);
    End;
  End;
  If (taste=#8) and (length(path)=0) Then Begin
    Sound(3000); Delay(3); NoSound; End;
  If not(taste in [#8,#13,#0]) And (x<=464) Then Begin
    SetColor(0);
    path:=path+taste;
    ShowText(taste);
    Inc(x,8);
  End;
Until taste=#13;
{$I-}
ChDir(path);
{$I+}
If (IOResult=0) And (length(path)<>0) Then Break;
Until False;
End;
{-----Suchpfad wechseln-----}
Procedure ChangePath;
Begin
  M.Show(False);
  Mnu.Win(210,154,480,174,13,14,15);
  GetPath(Path);
  Mnu.Win(210,154,480,174,15,14,13);
  SetColor(0);
  If length(path)<32 Then ShowTextXY(220,160,path)
    Else ShowTextxy(220,160,copy(path,length(path)-30,31));
  GetFiles(Path);
  Sortieren;
  filestart:=0;
  activef:=1; oldf:=1;

```

```

oldx:=170; oldy:=214;
WriteFileName(activex);
WriteFiles(1+filestart*24,24+filestart*24);
M.Show(True);
End;
{-----Prozedur für das Auswählen einer Datei mit der Maus-----}
Procedure PickFile(mx,my,mt:Integer;var load:Boolean);
Var OldActiveF: Byte;
    f24: Integer;
Begin
    OldActiveF:=ActiveF;
    f24:=filestart*24;
    If M.Innen(170,214,388,344,mx,my) and ((mt=linketaste) or (mt=rechtetaste)) Then Begin
        ActiveF:=1+f24+(my-213) div 11+12*((mx-169) div 110);
        If (ActiveF<=zv) And (mt=rechtetaste) And
            (copy(files[activex],length(files[activex]),1)<>'\'') Then
            Load:=True Else Load:=False;
        If (ActiveF<=zv) And (mt=rechtetaste) And
            (copy(files[activex],length(files[activex]),1)='\'') Then Begin
            M.Show(False);
            If ((oldf>=1+f24) And (oldf<=24+f24)) Then
                SetText(oldx,oldy,files[oldf],0,14);
            x:=170+((mx-169) div 110)*110;
            y:=214+((my-213) div 11)*11;
            ActiveF:=1+f24+(my-213) div 11+12*((mx-169) div 110);
            SetText(x,y,files[activex],15,0);
            WriteFileName(activex);
            path:=BackSlash(path)+copy(files[activex],1,length(files[activex])-1);
            ChDir(path); GetDir(0,path);
            GetFiles(path); Sortieren;
            filestart:=0;
            activex:=1; oldf:=1;
            oldx:=170; oldy:=214;
            Repeat
                M.status(mx,my,mt);
            Until mt=keinetaste;
            M.Show(False);
            Mnu.Win(210,154,480,174,15,14,13);
            SetColor(0);
            If length(path)<32 Then ShowTextXY(220,160,path)
                Else ShowTextXY(220,160,copy(path,length(path)-30,31));
            WriteFileName(activex);
            M.Show(True);
            WriteFiles(1+f24,24+f24);
        End;
    If (activex<>oldf) and (activex<=zv) Then Begin
        M.Show(False);
        If ((oldf>=1+f24) And (oldf<=24+f24)) Then
            SetText(oldx,oldy,files[oldf],0,14);
        x:=170+((mx-169) div 110)*110;
        y:=214+((my-213) div 11)*11;
        ActiveF:=1+f24+(my-213) div 11+12*((mx-169) div 110);
        SetText(x,y,files[activex],15,0);
    End;
End;

```



```

WriteFileName(activex);
M.Show(True);
oldf:=activex;
oldx:=x;
oldy:=y;
End Else ActiveF:=OldActiveF;
End;
End;
{----Ständige Abfrage des Mausstatus und Ausführung der Befehle----}
Procedure MausAbfrage;
var Ok, cancel, left, right: Boolean;
    okText: String;
Begin
left:=False; right:=False;
Ok:=False; cancel:=False;
If Typ Then okText:='LADEN' Else okText:='SPEICHERN';
Repeat
M.Status(mx,my,mt);
Mnu.GetButton(160,351,180,371,linketaste,mx,my,mt,#17,left);
Mnu.GetButton(370,351,390,371,linketaste,mx,my,mt,#16,right);
Mnu.GetButton(400,351,480,371,linketaste,mx,my,mt,'ABBRUCH',cancel);
Mnu.GetButton(400,321,480,341,linketaste,mx,my,mt,oktext,ok);
PickFile(mx,my,mt,ok);
If M.Innen(170,214,388,344,mx,my) Then M.SetKreuz
    Else M.SetHand;
If (mt=linketaste) and M.Innen(210,154,480,174,mx,my) Then ChangePath;
If (mt=linketaste) and M.Innen(210,125,324,145,mx,my) Then SearchFile(code,cancel);
If left Then Begin
left:=False;
If filestart>0 Then Begin
Dec(filestart);
WriteFiles(1+filestart*24,24+filestart*24);
End;
End;
If right Then Begin
right:=False;
If (24+filestart*24)<zv Then Begin
Inc(filestart);
WriteFiles(1+filestart*24,24+filestart*24);
End;
End;
If ok Then Begin
ok:=False;
If Typ Then LoadFile(files[activex],bob,code)
    Else SaveFile(files[activex],bob,code);
If code=False Then cancel:=True
    Else cancel:=False;
M.Show(True);
End;
Until cancel=True;
M.SetHand
End;
{----Hauptteil der Prozedure LadenSichernWindow----}

```

```

Begin
  M.Show(False);
  GetScreen;
  ScreenInit;
  filestart:=0;
  activef:=1; oldf:=1;
  oldx:=170; oldy:=214;
  code:=True;
  GetDir(0,path);
  SetColor(0);
  If length(path)<32 Then ShowTextXY(220,160,path)
    Else ShowTextXY(220,160,copy(path,length(path)-30,31));
  GetFiles(path);
  Sortieren;
  If typ Then writefilename(activef) Else
  If bob Then Begin
    activef:=241;
    files[activef]:='LIFE'+GetCountNumber+'.BLD';
    SetFileName(files[activef]);
  End Else Begin
    activef:=241;
    files[activef]:='LIFE'+GetCountNumber+'.KFG';
    SetFileName(files[activef]);
  End;
  WriteFiles(1+filestart*24,24+filestart*24);
  M.Show(True);
  MausAbfrage;
  M.Show(False);
  PutScreen;
  M.MWindow(0,0,maxX,maxY);
  M.Show(True);
End;
{----Konfigurationseditor----}
Procedure BEDINGUNGEN;
const exptext1: Array[1..6] Of String=('Anzahl der leb. Zellen',
  'Anzahl der blauen Zellen',
  'Anzahl der roten Zellen',
  'Anzahl der grünen Zellen',
  'Anzahl der toten Zellen',
  'Gesamtwert der Zellen');
  exptext2: Array[1..8] Of String=('= A','<> A','< A','> A',
  '> A AND < B','< A OR > B',
  'MOD A = B','ZIFSUM < A');

Var typ,oldtyp: Byte;
{----Variablen des Konfigurationseditor werden initialisiert----}
Procedure VarInit;
Begin
  typ:=1;
  oldtyp:=1;
End;
{----Würfel zeichnen----}
Procedure SetCube(x,y,sk,c1,c2:Integer);
Var p: Array[1..3,1..5] Of PointType;

```

```

Begin
  p[1,1].x:=x;    p[1,1].y:=y;
  p[1,2].x:=x+3*sk;  p[1,2].y:=y+sk;
  p[1,3].x:=x;    p[1,3].y:=y+2*sk;
  p[1,4].x:=x-3*sk;  p[1,4].y:=y+sk;
  p[1,5]:=p[1,1];
  p[2,1]:=p[1,4];
  p[2,2].x:=x-3*sk;  p[2,2].y:=y+4*sk;
  p[2,3].x:=x;    p[2,3].y:=y+5*sk;
  p[2,4]:=p[1,3];
  p[2,5]:=p[2,1];
  p[3,1]:=p[2,4];
  p[3,2]:=p[1,2];
  p[3,3].x:=x+3*sk;  p[3,3].y:=y+4*sk;
  p[3,4]:=p[2,3];
  p[3,5]:=p[3,1];
  SetFillStyle(1,c2);
  SetColor(c1);
  Fillpoly(5,p[1]);
  FillPoly(5,p[2]);
  FillPoly(5,p[3]);
End;
{----Kreuz zeichnen----}
Procedure ZeichneKreuz(x,y:Integer);
Begin
  SetFillStyle(1,0);
  SetColor(0);
  Bar(x-2,y,x+2,y+30);
  Bar(x-10,y+8,x+10,y+13);
End;
{----Text schreiben bei Änderung der Priorität----}
Procedure DrawPrior(typ,cell:Byte);
Var c:Integer;
Begin
  c:=(cell-1)*79;
  SetFillStyle(1,14);
  SetColor(14);
  SolidBar(72,136+c,384,144+c);
  SolidBar(72,187+c,152,195+c);
  SolidBar(16,136+c,52,170+c);
  Case Cvar[typ,cell].prior of
  0:Begin
    ZeichneKreuz(34,136+c);
    SetText(72,136+c,'Die Zelle stirbt, wenn:',0,15);
    SetText(72,187+c,'Priorität:',0,15);
  End;
  1:Begin
    SetCube(34,136+c,5,1,3);
    SetText(72,136+c,'Zelle wird zu einer blauen Zelle, wenn:',3,15);
    SetText(72,187+c,'Priorität:',3,15);
  End;
  2:Begin
    SetCube(34,136+c,5,4,6);

```

```

    SetText(72,136+c,'Zelle wird zu einer roten Zelle, wenn:',6,15);
    SetText(72,187+c,'Priorität:',6,15);
End;
3:Begin
    SetCube(34,136+c,5,7,8);
    SetText(72,136+c,'Zelle wird zu einer grünen Zelle, wenn:',7,15);
    SetText(72,187+c,'Priorität:',7,15);
End;
End;
End;
{-----Prioritäten ändern-----}
Procedure ChangePrior(typ,cell:Byte);
Var h1,i: Byte;
Begin
    h1:=Cvar[typ,cell].prior;
    If Cvar[typ,cell].prior<3 Then Inc(Cvar[typ,cell].prior)
        Else Cvar[typ,cell].prior:=0;
    For i:= 1 to 4 do
        If i<>cell Then
            If Cvar[typ,i].prior=Cvar[typ,cell].prior Then Begin
                Cvar[typ,i].prior:=h1;
                break;
            End;
        M.Show(False);
        DrawPrior(typ,cell);
        DrawPrior(typ,i);
        M.Show(True);
    End;
{-----Prozeduren für Darstellung der Konfiguration am Bildschirm-----}
Procedure WriteConfig(typ,cell:Byte);
Begin
    M.Show(False);
    SetFillStyle(1,14);
    SetColor(14);
    SolidBar(72,156+(cell-1)*79,404,176+(cell-1)*79);
    If Cvar[typ,cell].CellSwch Then
        Mnu.ButtonOut(72,156+(cell-1)*79,271,176+(cell-1)*79,exptext1[Cvar[typ,cell].exp1]);
    M.Show(True);
End;
Function WriteConfigNumber(typ,cell:Byte):String;
Var st,sta,stb: String;
Begin
    If (Cvar[typ,cell].exp3[1]=0) And (Cvar[typ,cell].exp2=7) Then
        Cvar[typ,cell].exp3[1]:=1;
    str(Cvar[typ,cell].exp3[1]:2,sta);
    str(Cvar[typ,cell].exp3[2]:2,stb);
    case Cvar[typ,cell].exp2 of
        1:st:= ' '+sta;
        2:st:= '<>' +sta;
        3:st:= '<' +sta;
        4:st:= '>' +sta;
        5:st:= '>' +sta+' AND <' +stb;
        6:st:= '<' +sta+' OR >' +stb;
    end;

```

```

7:st:='MOD '+sta+' = '+stb;
8:st:='ZIFSUM < '+sta;
End;
WriteConfigNumber:=st;
End;
{----Darstellen aller aktiven Zellregeln am Bildschirm----}
Procedure WriteConfigforall(typ:Byte);
Var i: Integer;
Begin
For i:= 1 To 4 do Begin
SetFillStyle(1,14);
SetColor(14);
SolidBar(72,156+(i-1)*79,404,176+(i-1)*79);
If Cvar[typ,i].CellSwtch Then Begin
Mnu.ButtonOut(72,156+(i-1)*79,271,176+(i-1)*79,exptext1[Cvar[typ,i].exp1]);
Mnu.ButtonOut(275,156+(i-1)*79,400,176+(i-1)*79,WriteConfigNumber(typ,i));
End;
DrawPrior(typ,i);
End;
End;
{----Darstellen der Ein/Aus-Schaltflächen----}
Procedure SetSwitches(akt:Byte);
Begin
If Cvar[akt,1].CellSwtch Then Mnu.ButtonIn(15,176,53,194,'EIN')
Else Mnu.ButtonOut(15,176,53,194,'AUS');
If Cvar[akt,2].CellSwtch Then Mnu.ButtonIn(15,255,53,273,'EIN')
Else Mnu.ButtonOut(15,255,53,273,'AUS');
If Cvar[akt,3].CellSwtch Then Mnu.ButtonIn(15,334,53,352,'EIN')
Else Mnu.ButtonOut(15,334,53,352,'AUS');
If Cvar[akt,4].CellSwtch Then Mnu.ButtonIn(15,413,53,431,'EIN')
Else Mnu.ButtonOut(15,413,53,431,'AUS');
End;
{----Werte der Zellen schreiben----}
Procedure WriteCellValues;
Var st: String;
Begin
Str(Cvar[1,1].wert,st); Mnu.ButtonIn(580,391,596,407,st);
Str(Cvar[1,2].wert,st); Mnu.ButtonIn(580,411,596,427,st);
Str(Cvar[1,3].wert,st); Mnu.ButtonIn(580,431,596,447,st);
Str(Cvar[1,4].wert,st); Mnu.ButtonIn(580,451,596,467,st);
End;
{----Momentane Anzahl der Zellregeln schreiben----}
Procedure WriteActiveCellRules;
Var st: String;
i,j: Integer;
z: Byte;
Begin
z:=0;
For i:= 1 To 5 Do
For j:= 1 To 4 Do
If Cvar[i,j].CellSwtch Then Inc(z);
SetFillStyle(1,14);
SetColor(14);

```

```

SolidBar(397,456,413,464);
Str(z:2,st);
SetText(397,456,st,15,0);
End;
{-----Aufbau und Zeichnen des Konfigurationsfenster-----}
Procedure Init;
Begin
Mnu.Win(0,0,maxX,maxY,15,14,13); Mnu.Win(4,4,maxX-4,25,13,14,15);
Mnu.Win(4,104,416,444,15,14,13); Mnu.Win(8,108,412,124,13,14,15);
Mnu.Win(8,128,60,203,13,14,15); Mnu.Win(64,128,412,203,13,14,15);
Mnu.Win(8,207,60,282,13,14,15); Mnu.Win(64,207,412,282,13,14,15);
Mnu.Win(8,286,60,361,13,14,15); Mnu.Win(64,286,412,361,13,14,15);
Mnu.Win(8,365,60,440,13,14,15); Mnu.Win(64,365,412,440,13,14,15);
Mnu.Win(420,29,635,366,15,14,13); Mnu.Win(424,33,631,49,13,14,15);
Mnu.Win(424,53,631,202,13,14,15); Mnu.Win(424,206,631,282,13,14,15);
Mnu.Win(424,286,631,362,13,14,15);Mnu.Win(4,29,83,100,13,14,15);
Mnu.Win(87,29,166,100,13,14,15); Mnu.Win(170,29,249,100,13,14,15);
Mnu.Win(253,29,332,100,13,14,15); Mnu.Win(336,29,416,100,13,14,15);
Mnu.ButtonOut(88,30,165,99,""); Mnu.ButtonOut(171,30,248,99,"");
Mnu.ButtonOut(254,30,331,99,""); Mnu.ButtonOut(337,30,415,99,"");
Mnu.Win(420,370,635,475,15,14,13);Mnu.Win(424,374,631,471,13,14,15);
Mnu.Win(4,448,276,475,13,14,15);
Mnu.ButtonOut(160,183,176,199,'1'); Mnu.ButtonOut(160,262,176,278,'2');
Mnu.ButtonOut(160,341,176,357,'3'); Mnu.ButtonOut(160,420,176,436,'4');
Mnu.ButtonOut(8,452,68,471,'Fertig');
Mnu.ButtonOut(72,452,162,471,'Speichern');
Mnu.ButtonOut(166,452,228,471,'Laden');
Mnu.ButtonOut(232,452,272,471,'Neu');
SetText(282,456,'Aktive Regeln:',0,15);
WriteActiveCellRules;
Mnu.ButtonOut(560,391,576,407,'+'); Mnu.ButtonOut(560,411,576,427,'+');
Mnu.ButtonOut(560,431,576,447,'+'); Mnu.ButtonOut(560,451,576,467,'+');
WriteCellValues;
Mnu.ButtonIn(14,175,54,195,""); Mnu.ButtonIn(14,254,54,274,"");
Mnu.ButtonIn(14,333,54,353,""); Mnu.ButtonIn(14,412,54,432,"");
Mnu.ButtonOut(600,391,616,407,'-'); Mnu.ButtonOut(600,411,616,427,'-');
Mnu.ButtonOut(600,431,616,447,'-'); Mnu.ButtonOut(600,451,616,467,'-');
Mnu.ButtonOut(428,57,627,77,'Anzahl der leb. Zellen');
Mnu.ButtonOut(428,81,627,101,'Anzahl der blauen Zellen');
Mnu.ButtonOut(428,105,627,125,'Anzahl der roten Zellen');
Mnu.ButtonOut(428,129,627,149,'Anzahl der grünen Zellen');
Mnu.ButtonOut(428,153,627,173,'Anzahl der toten Zellen');
Mnu.ButtonOut(428,177,627,198,'Gesamtwert der Zellen');
Mnu.ButtonOut(428,210,475,230,exptext2[1]);
Mnu.ButtonOut(479,210,526,230,exptext2[2]);
Mnu.ButtonOut(530,210,577,230,exptext2[3]);
Mnu.ButtonOut(581,210,627,230,exptext2[4]);
Mnu.ButtonOut(428,234,526,254,exptext2[5]);
Mnu.ButtonOut(530,234,627,254,exptext2[6]);
Mnu.ButtonOut(428,258,526,278,exptext2[7]);
Mnu.ButtonOut(530,258,627,278,exptext2[8]);
Mnu.ButtonOut(428,290,448,310,'7'); Mnu.ButtonOut(452,290,472,310,'8');
Mnu.ButtonOut(476,290,496,310,'9'); Mnu.ButtonOut(500,290,520,310,'0');

```

```

Mnu.ButtonOut(428,314,448,334,'4'); Mnu.ButtonOut(452,314,472,334,'5');
Mnu.ButtonOut(476,314,496,334,'6'); Mnu.ButtonOut(500,314,520,334,'A');
Mnu.ButtonOut(428,338,448,358,'1'); Mnu.ButtonOut(452,338,472,358,'2');
Mnu.ButtonOut(476,338,496,358,'3'); Mnu.ButtonOut(500,338,520,358,'B');
Mnu.ButtonIn(550,290,607,310,'A: ');Mnu.ButtonIn(550,314,607,334,'B: ');
Mnu.ButtonOut(550,338,607,358,'Ok');
SetSwitches(typ);
Mnu.ButtonIn(5,30,82,99,'Global');
SetText(335+(80-textwidth('Tote')) div 2,53,'Tote',15,0);
SetText(335+(80-textwidth('Zelle')) div 2,69,'Zelle',15,0);
SetText((maxX-4-textwidth(mtext[4])) div 2,11,mtext[4],15,0);
SetText(4+(412-textwidth('Konfiguration für alle Zellen')) div 2,112,
'Konfiguration für alle Zellen',15,0);
SetText(424+(215-textwidth('Ausdrücke')) div 2,37,'Ausdrücke',15,0);
SetText(424+(207-textwidth('WERTIGKEIT DER ZELLEN')) div 2,378,
'Wertigkeit der Zellen',15,0);
SetText(432,395,'Tote Zellen:',0,15);
SetText(432,415,'Blaue Zellen:',3,15);
SetText(432,435,'Rote Zellen:',6,15);
SetText(432,455,'Grüne Zellen:',7,15);
SetCube(127,49,6,3,1);
SetCube(210,49,6,6,4);
SetCube(293,49,6,9,7);
WriteConfigForAll(typ);
End;
{-----Ständige Abfrage des Mausstatus und Ausführen der Befehle-----}
Procedure MausAbfrage;
Var Ende,blplus,blminus,rtplus,rtminus: Boolean;
    grplus,grminus,totplus,totminus: Boolean;
    Load,Save,Neu,Code: Boolean;
    swtchk,prd: Array[1..4] Of Boolean;
    korrchk: Array[1..4,1..2] Of Boolean;
    i,j: Integer;

{-----Wertigkeit der Zellen ändern-----}
Procedure ChangeValue;
Var St: String;
Begin
    If totplus Then Begin
        totplus:=False;
        If (Cvar[1,1].wert<9) Then Begin
            Inc(Cvar[1,1].wert); Str(Cvar[1,1].wert,st);
            M.Show(False); Mnu.ButtonIn(580,391,596,407,st); M.Show(True)
        End;
    End;
    If totminus Then Begin
        totminus:=False;
        If (Cvar[1,1].wert>0) Then Begin
            Dec(Cvar[1,1].wert); Str(Cvar[1,1].wert,st);
            M.Show(False); Mnu.ButtonIn(580,391,596,407,st); M.Show(True)
        End;
    End;
    If blplus Then Begin

```

```

bplus:=False;
If (Cvar[1,2].wert<9) Then Begin
  Inc(Cvar[1,2].wert); Str(Cvar[1,2].wert,st);
  M.Show(False); Mnu.ButtonIn(580,411,596,427,st); M.Show(True)
End;
End;
If blminus Then Begin
  blminus:=False;
  If (Cvar[1,2].wert>0) Then Begin
    Dec(Cvar[1,2].wert); Str(Cvar[1,2].wert,st);
    M.Show(False); Mnu.ButtonIn(580,411,596,427,st); M.Show(True)
  End;
End;
If rtplus Then Begin
  rtplus:=False;
  If (Cvar[1,3].wert<9) Then Begin
    Inc(Cvar[1,3].wert); Str(Cvar[1,3].wert,st);
    M.Show(False); Mnu.ButtonIn(580,431,596,447,st); M.Show(True)
  End;
End;
If rtminus Then Begin
  rtminus:=False;
  If (Cvar[1,3].wert>0) Then Begin
    Dec(Cvar[1,3].wert); Str(Cvar[1,3].wert,st);
    M.Show(False); Mnu.ButtonIn(580,431,596,447,st); M.Show(True)
  End;
End;
If grplus Then Begin
  grplus:=False;
  If (Cvar[1,4].wert<9) Then Begin
    Inc(Cvar[1,4].wert); Str(Cvar[1,4].wert,st);
    M.Show(False); Mnu.ButtonIn(580,451,596,467,st); M.Show(True)
  End;
End;
If grminus Then Begin
  grminus:=False;
  If (Cvar[1,4].wert>0) Then Begin
    Dec(Cvar[1,4].wert); Str(Cvar[1,4].wert,st);
    M.Show(False); Mnu.ButtonIn(580,451,596,467,st); M.Show(True)
  End;
End;
End;
{----Prozedur für die Eingabe des 1.Teils der Regeln----}
Procedure GetExp1(typ,cell:Byte);
Var exp: Array[1..6] Of Boolean;
  Ende: Boolean;
  i: Integer;
Begin
  M.MWindow(424,53,631,202);
  For i:= 1 To 6 Do
    exp[i]:=False;
  Ende:=False;
  Repeat

```



```

M.Status(mx,my,mt);
Mnu.GetButton(428,57,627,77,linketaste,mx,my,mt,exptext1[1],exp[1]);
Mnu.GetButton(428,81,627,101,linketaste,mx,my,mt,exptext1[2],exp[2]);
Mnu.GetButton(428,105,627,125,linketaste,mx,my,mt,exptext1[3],exp[3]);
Mnu.GetButton(428,129,627,149,linketaste,mx,my,mt,exptext1[4],exp[4]);
Mnu.GetButton(428,153,627,173,linketaste,mx,my,mt,exptext1[5],exp[5]);
Mnu.GetButton(428,177,627,198,linketaste,mx,my,mt,exptext1[6],exp[6]);
For i:= 1 To 6 Do
  If exp[i] Then Begin
    Ende:=True;
    Break;
  End;
Until Ende;
Cvar[typ,cell].exp1:=i;
M.Show(False);
Mnu.ButtonOut(72,156+(cell-1)*79,271,176+(cell-1)*79,exptext1[i]);
M.Show(True);
M.MWindow(0,0,maxx,maxy);
End;
{----Prozedur für die Eingabe des 2.Teils der Regeln----}
Procedure GetExp2(typ,cell:Byte);
Var exp: Array[1..8] Of Boolean;
  Ende: Boolean;
  i: Integer;
Begin
  M.Mwindow(424,206,631,282);
  For i:= 1 To 8 Do
    exp[i]:=False;
  Ende:=False;
  Repeat
    M.Status(mx,my,mt);
    Mnu.GetButton(428,210,475,230,linketaste,mx,my,mt,exptext2[1],exp[1]);
    Mnu.GetButton(479,210,526,230,linketaste,mx,my,mt,exptext2[2],exp[2]);
    Mnu.GetButton(530,210,577,230,linketaste,mx,my,mt,exptext2[3],exp[3]);
    Mnu.GetButton(581,210,627,230,linketaste,mx,my,mt,exptext2[4],exp[4]);
    Mnu.GetButton(428,234,526,254,linketaste,mx,my,mt,exptext2[5],exp[5]);
    Mnu.GetButton(530,234,627,254,linketaste,mx,my,mt,exptext2[6],exp[6]);
    Mnu.GetButton(428,258,526,278,linketaste,mx,my,mt,exptext2[7],exp[7]);
    Mnu.GetButton(530,258,627,278,linketaste,mx,my,mt,exptext2[8],exp[8]);
  For i:= 1 to 8 do
    If exp[i] Then Begin
      Ende:=True;
      Break;
    End;
  Until Ende;
  Cvar[typ,cell].exp2:=i;
  M.Mwindow(0,0,maxx,maxy);
End;
{----Prozedur für die Eingabe des 3.Teils der Regeln----}
Procedure GetExp3(typ,cell:Byte);
Var prd: Array[1..12] Of Boolean;
  Ende,aob: Boolean;
  i,code: Integer;

```

```

sta, stb, zahl: Byte;
st: String;

Function LeadingZero(w: Byte): String;
Var s: String;
Begin
  Str(w: 0, s);
  If Length(s) = 1 Then s := '0' + s;
  LeadingZero := s;
End;
Procedure WriteNumber(zahl: String);
Begin
  M.Show(False);
  If aob Then Begin
    SetFillStyle(1, 14); SetColor(14);
    SolidBar(582, 297, 600, 305);
    SetText(582, 297, zahl, 15, 0)
  End Else Begin
    SetFillStyle(1, 14); SetColor(14);
    SolidBar(582, 321, 600, 329);
    SetText(582, 321, zahl, 15, 0);
  End;
  M.Show(True);
End;

Begin
  M.Mwindow(424, 286, 631, 362);
  Ende := False;
  zahl := 0;
  Cvar[typ, cell].exp3[1] := 0;
  Cvar[typ, cell].exp3[2] := 0;
  sta := Cvar[typ, cell].exp3[1];
  stb := Cvar[typ, cell].exp3[2];
  aob := False;
  WriteNumber(LeadingZero(sta));
  aob := True;
  WriteNumber(LeadingZero(stb));
  For i := 1 To 12 Do
    prd[i] := False;
  Repeat
    M.Status(mx, my, mt);
    Mnu.GetButton(428, 290, 448, 310, linketaste, mx, my, mt, '7', prd[1]);
    Mnu.GetButton(452, 290, 472, 310, linketaste, mx, my, mt, '8', prd[2]);
    Mnu.GetButton(476, 290, 496, 310, linketaste, mx, my, mt, '9', prd[3]);
    Mnu.GetButton(500, 290, 520, 310, linketaste, mx, my, mt, '0', prd[4]);
    Mnu.GetButton(428, 314, 448, 334, linketaste, mx, my, mt, '4', prd[5]);
    Mnu.GetButton(452, 314, 472, 334, linketaste, mx, my, mt, '5', prd[6]);
    Mnu.GetButton(476, 314, 496, 334, linketaste, mx, my, mt, '6', prd[7]);
    Mnu.GetButton(500, 314, 520, 334, linketaste, mx, my, mt, 'A', prd[8]);
    Mnu.GetButton(428, 338, 448, 358, linketaste, mx, my, mt, '1', prd[9]);
    Mnu.GetButton(452, 338, 472, 358, linketaste, mx, my, mt, '2', prd[10]);
    Mnu.GetButton(476, 338, 496, 358, linketaste, mx, my, mt, '3', prd[11]);
    Mnu.GetButton(500, 338, 520, 358, linketaste, mx, my, mt, 'B', prd[12]);
  Until Ende;
End;

```

```

Mnu.GetButton(550,338,607,358,linketaste,mx,my,mt,'Ok',Ende);
If prd[8] Then Begin
  prd[8]:=False;
  If aob=False Then Begin
    aob:=True;
    stb:=zahl;
    zahl:=sta;
    WriteNumber(leadingzero(zahl));
  End;
End;
If prd[12] Then Begin
  prd[12]:=False;
  If aob Then Begin
    aob:=False;
    sta:=zahl;
    zahl:=stb;
    WriteNumber(leadingzero(zahl));
  End;
End;
If prd[1] Then Begin
  prd[1]:=False;
  str(zahl:2,st);
  st:=copy(st,2,1)+'7';
  val(st,zahl,code);
  WriteNumber(leadingzero(zahl));
End;
If prd[2] Then Begin
  prd[2]:=False;
  str(zahl:2,st);
  st:=copy(st,2,1)+'8';
  val(st,zahl,code);
  WriteNumber(leadingzero(zahl));
End;
If prd[3] Then Begin
  prd[3]:=False;
  str(zahl:2,st);
  st:=copy(st,2,1)+'9';
  val(st,zahl,code);
  WriteNumber(leadingzero(zahl));
End;
If prd[4] Then Begin
  prd[4]:=False;
  str(zahl:2,st);
  st:=copy(st,2,1)+'0';
  val(st,zahl,code);
  WriteNumber(leadingzero(zahl));
End;
If prd[5] Then Begin
  prd[5]:=False;
  str(zahl:2,st);
  st:=copy(st,2,1)+'4';
  val(st,zahl,code);
  WriteNumber(leadingzero(zahl));

```

```

End;
If prd[6] Then Begin
  prd[6]:=False;
  str(zahl:2,st);
  st:=copy(st,2,1)+'5';
  val(st,zahl,code);
  WriteNumber(leadingzero(zahl));
End;
If prd[7] Then Begin
  prd[7]:=False;
  str(zahl:2,st);
  st:=copy(st,2,1)+'6';
  val(st,zahl,code);
  WriteNumber(leadingzero(zahl));
End;
If prd[9] Then Begin
  prd[9]:=False;
  str(zahl:2,st);
  st:=copy(st,2,1)+'1';
  val(st,zahl,code);
  WriteNumber(leadingzero(zahl));
End;
If prd[10] Then Begin
  prd[10]:=False;
  str(zahl:2,st);
  st:=copy(st,2,1)+'2';
  val(st,zahl,code);
  WriteNumber(leadingzero(zahl));
End;
If prd[11] Then Begin
  prd[11]:=False;
  str(zahl:2,st);
  st:=copy(st,2,1)+'3';
  val(st,zahl,code);
  WriteNumber(leadingzero(zahl));
End;
Until Ende;
If aob Then sta:=zahl Else stb:=zahl;
Cvar[typ,cell].exp3[1]:=sta;
Cvar[typ,cell].exp3[2]:=stb;
M.Show(False);
SetFillStyle(1,14);SetColor(14);
SolidBar(582,297,600,305);
SolidBar(582,321,600,329);
Mnu.ButtonOut(275,156+(cell-1)*79,400,176+(cell-1)*79,WriteConfigNumber(typ,cell));
M.Show(True);
M.Mwindow(0,0,maxx,maxy);
End;
{-----Eingabe der Regeln-----}
Procedure GetConfig(typ,cell:Byte);
Begin
  GetExp1(typ,cell);
  GetExp2(typ,cell);

```

```

GetExp3(typ,cell);
End;
{----Überschrift des Regelfensters ändern----}
Procedure SetWindow(typ:Byte);
Begin
  M.Show(False);
  SetFillstyle(1,14);
  SetColor(14);
  SolidBar(12,112,408,120);
  case typ of
    1:Begin
      SetText(4+(412-textwidth('Konfiguration für alle Zellen')) div 2,112,
        'Konfiguration für alle Zellen',15,0);
    End;
    2:Begin
      SetText(4+(412-textwidth('Konfiguration für die blauen Zellen')) div 2,112,
        'Konfiguration für die blauen Zellen',3,15);
    End;
    3:Begin
      SetText(4+(412-textwidth('Konfiguration für die roten Zellen')) div 2,112,
        'Konfiguration für die roten Zellen',6,15);
    End;
    4:Begin
      SetText(4+(412-textwidth('Konfiguration für die grünen Zellen')) div 2,112,
        'Konfiguration für die grünen Zellen',7,15);
    End;
    5:Begin
      SetText(4+(412-textwidth('Konfiguration für die toten Zellen')) div 2,112,
        'Konfiguration für die toten Zellen',0,15);
    End;
  End;
  SetSwitches(typ);
  WriteConfigForAll(typ);
  M.Show(True);
End;
{----Setzen der Zellschaltflächen----}
Procedure SetButtons;
Begin
  M.Show(False);
  case oldtyp of
    1:Mnu.ButtonOut(5,30,82,99,'Global');
    2:Begin Mnu.ButtonOut(88,30,165,99,"");SetCube(127,49,6,3,1); End;
    3:Begin Mnu.ButtonOut(171,30,248,99,"");SetCube(210,49,6,6,4); End;
    4:Begin Mnu.ButtonOut(254,30,331,99,"");SetCube(293,49,6,9,7); End;
    5:Begin
      Mnu.ButtonOut(337,30,415,99,"");
      SetText(335+(80-textwidth('Tote')) div 2,53,'Tote',15,0);
      SetText(335+(80-textwidth('Zelle')) div 2,69,'Zelle',15,0);
    End;
  End;
  case typ of
    1:Mnu.ButtonIn(5,30,82,99,'Global');
    2:Begin Mnu.ButtonIn(88,30,165,99,"");SetCube(127,45,8,1,3); End;
  End;

```

```

3:Begin Mnu.ButtonIn(171,30,248,99,"");SetCube(210,45,8,4,6); End;
4:Begin Mnu.ButtonIn(254,30,331,99,"");SetCube(293,45,8,7,9) End;
5:Begin
  Mnu.ButtonIn(337,30,415,99,"");
  SetText(335+(80-textwidth('Tote')) div 2+1,54,'Tote',0,15);
  SetText(335+(80-textwidth('Zelle')) div 2+1,70,'Zelle',0,15);
End;
End;
M.Show(True);
End;
{-----Abfrage, ob Zellschaltflächen gedrückt wurden-----}
Procedure ZellTypAbfrage(mx,my,mt:Integer);
Begin
  If mt=linketaste Then Begin
    If M.Innen(4,29,83,100,mx,my) and (typ<>1) Then Begin
      typ:=1; SetButtons; oldtyp:=1;
      SetWindow(typ);
      Repeat
        M.Status(mx,my,mt);
      Until mt=keinetaste;
    End;
    If M.Innen(87,29,166,100,mx,my) and (typ<>2) Then Begin
      typ:=2; SetButtons; oldtyp:=2;
      SetWindow(typ);
      Repeat
        M.Status(mx,my,mt);
      Until mt=keinetaste;
    End;
    If M.Innen(170,29,249,100,mx,my) and (typ<>3) Then Begin
      typ:=3; setButtons; oldtyp:=3;
      SetWindow(typ);
      Repeat
        M.Status(mx,my,mt);
      Until mt=keinetaste;
    End;
    If M.Innen(253,29,332,100,mx,my) and (typ<>4) Then Begin
      typ:=4; setbuttons; oldtyp:=4;
      SetWindow(typ);
      Repeat
        M.Status(mx,my,mt);
      Until mt=keinetaste;
    End;
    If M.Innen(336,29,416,100,mx,my) and (typ<>5) Then Begin
      typ:=5; setbuttons; oldtyp:=5;
      SetWindow(typ);
      Repeat
        M.Status(mx,my,mt);
      Until mt=keinetaste;
    End;
  End;
End;
End;
{-----Hauptteil der Prozedur Mausabfrage-----}
Begin

```

```

Ende:=False; Blplus:=False; Blminus:=False; Rtplus:=False;
Rtminus:=False; Grplus:=False; Grminus:=False; totplus:=False;
totminus:=False; Neu:=False; Load:=False; Save:=False;
For i:= 1 To 4 Do Begin
  swtchk[i]:=False;
  prd[i]:=False;
  For j:= 1 To 2 Do
    korrchk[i,j]:=False;
End;
Repeat
  M.Status(mx,my,mt);
  Mnu.GetButton(8,452,68,471,linketaste,mx,my,mt,'Fertig',Ende);
  Mnu.GetButton(72,452,162,471,linketaste,mx,my,mt,'Speichern',Save);
  Mnu.GetButton(166,452,228,471,linketaste,mx,my,mt,'Laden',load);
  Mnu.GetButton(232,452,272,471,linketaste,mx,my,mt,'Neu',neu);
  Mnu.GetSwitch(14,175,54,195,linketaste,mx,my,mt,'EIN','AUS',Cvar[typ,1].CellSwtch,swtchk[1]);
  Mnu.GetSwitch(14,254,54,274,linketaste,mx,my,mt,'EIN','AUS',Cvar[typ,2].CellSwtch,swtchk[2]);
  Mnu.GetSwitch(14,333,54,353,linketaste,mx,my,mt,'EIN','AUS',Cvar[typ,3].CellSwtch,swtchk[3]);
  Mnu.GetSwitch(14,412,54,432,linketaste,mx,my,mt,'EIN','AUS',Cvar[typ,4].CellSwtch,swtchk[4]);
  ZellTypAbfrage(mx,my,mt);
  Mnu.GetButton(560,391,576,407,linketaste,mx,my,mt,'+',totplus);
  Mnu.GetButton(560,411,576,427,linketaste,mx,my,mt,'+',blplus);
  Mnu.GetButton(560,431,576,447,linketaste,mx,my,mt,'+',rtplus);
  Mnu.GetButton(560,451,576,467,linketaste,mx,my,mt,'+',grplus);
  Mnu.GetButton(600,391,616,407,linketaste,mx,my,mt,'-',totminus);
  Mnu.GetButton(600,411,616,427,linketaste,mx,my,mt,'-',blminus);
  Mnu.GetButton(600,431,616,447,linketaste,mx,my,mt,'-',rtminus);
  Mnu.GetButton(600,451,616,467,linketaste,mx,my,mt,'-',grminus);
  If Cvar[typ,1].CellSwtch Then Begin
    Mnu.GetButton(72,156,271,176,linketaste,mx,my,mt,exptext1[Cvar[typ,1].exp1],korrchk[1,1]);
    Mnu.GetButton(275,156,400,176,linketaste,mx,my,mt,writeconfignumber(typ,1),korrchk[1,2]);
  End;
  If Cvar[typ,2].CellSwtch Then Begin
    Mnu.GetButton(72,235,271,255,linketaste,mx,my,mt,exptext1[Cvar[typ,2].exp1],korrchk[2,1]);
    Mnu.GetButton(275,235,400,255,linketaste,mx,my,mt,writeconfignumber(typ,2),korrchk[2,2]);
  End;
  If Cvar[typ,3].CellSwtch Then Begin
    Mnu.GetButton(72,314,271,334,linketaste,mx,my,mt,exptext1[Cvar[typ,3].exp1],korrchk[3,1]);
    Mnu.GetButton(275,314,400,334,linketaste,mx,my,mt,writeconfignumber(typ,3),korrchk[3,2]);
  End;
  If Cvar[typ,4].CellSwtch Then Begin
    Mnu.GetButton(72,393,271,413,linketaste,mx,my,mt,exptext1[Cvar[typ,4].exp1],korrchk[4,1]);
    Mnu.GetButton(275,393,400,413,linketaste,mx,my,mt,writeconfignumber(typ,4),korrchk[4,2]);
  End;
  Mnu.GetButton(160,183,176,199,linketaste,mx,my,mt,'1',prd[1]);
  Mnu.GetButton(160,262,176,278,linketaste,mx,my,mt,'2',prd[2]);
  Mnu.GetButton(160,341,176,357,linketaste,mx,my,mt,'3',prd[3]);
  Mnu.GetButton(160,420,176,436,linketaste,mx,my,mt,'4',prd[4]);
  If Neu Then Begin
    Neu:=False;
    ResetVar;
    M.Show(False);
    SetSwitches(typ);
  End;
End;

```

```

WriteConfigForAll(typ);
WriteActiveCellRules;
M.Show(True);
End Else
If Load Then Begin
Load:=False;
LadenSichernWindow(True,False,code);
If code=False Then Begin
M.Show(False);
SetSwitches(typ);
WriteConfigForAll(typ);
WriteCellValues;
M.Show(True);
End;
End Else
If Save Then Begin
save:=False;
LadenSichernWindow(False,False,code);
End Else
For i:= 1 To 4 Do Begin
If swtchk[i] Then Begin
swtchk[i]:=False;
If Cvar[typ,i].CellSwtch Then GetConfig(typ,i)
Else WriteConfig(typ,i);
WriteActiveCellRules;
End;
If prd[i] Then Begin
prd[i]:=False;
ChangePrior(typ,i);
End;
If korrchk[i,1] Then Begin
korrchk[i,1]:=False;
GetExp1(typ,i);
End;
If korrchk[i,2] then Begin
korrchk[i,2]:=False;
GetExp2(typ,i);
GetExp3(typ,i);
End;
End;
ChangeValue;
Until Ende;
End;
{-----Hauptteil der Prozedur Konfiguration-----}
Begin
M.Show(False);
GetScreen;
VarInIt;
Init;
M.Show(True);
MausAbfrage;
M.Show(False);
PutScreen;

```



```

    RasterZeichnen(True);
    M.Show(True);
End;
{-----Zelleditor-----}
Procedure BEARBEITEN;
Var oldux,olduy,olddrehw,oldkippw,oldDistanz,oldZoom_Faktor: Integer;
    oldsp: Shortint;
    ab,color,color2: Integer;
    level,koordtyp: Byte;
    oldRahmenChk: Boolean;

Procedure RadioButtons(typ:Byte); forward;
Procedure ColorButtons(color,color2:Byte); forward;
{-----ZellDatum schreiben-----}
Procedure WriteDate(Datum:Integer);
Var st: String;
Begin
    SetFillStyle(1,14);
    SetColor(14);
    SolidBar(68,11,108,19);
    str(Datum,st);
    SetText(68,11,st,15,0);
End;
{-----Initialisierung der Variablen und Bildschirmaufbau des Editors-----}
Procedure Init;
Begin
    Level:=1;
    KoordTyp:=1;
    Color:=1; Color2:=0;
    Mnu.Win(0,0,maxX,maxY,15,14,13); Mnu.Win(4,4,420,25,13,14,15);
    Mnu.Win(4,29,420,444,13,14,15); Mnu.Win(424,4,635,250,13,14,15);
    Mnu.Win(424,254,635,475,13,14,15); Mnu.Win(4,448,420,475,13,14,15);
    Mnu.ButtonOut(454,220,605,240,'Aktualisieren');
    SetText((404-textwidth(mtext[3])) div 2,11,mtext[3],15,0);
    SetText(12,11,'Datum:',0,15);
    WriteDate(Datum);
    SetText(434,274,'Zellenanzahl X:',0,15);
    SetText(434,294,'Zellenanzahl Y:',0,15);
    SetText(434,314,'Zellenanzahl Z:',0,15);
    SetText(434,340,'X,Y Koordinatensystem',0,15);
    SetText(434,360,'X,Z Koordinatensystem',0,15);
    SetText(434,380,'Y,Z Koordinatensystem',0,15);
    SetText(434,406,'Ebene: von',0,15);
    SetText(342,11,'X:',0,15);
    SetText(382,11,'Y:',0,15);
    Mnu.Win(434,428,480,458,15,14,13); Mnu.Win(482,428,528,458,3,2,1);
    Mnu.Win(530,428,576,458,6,5,4); Mnu.Win(578,428,624,458,9,8,7);
    Mnu.ButtonOut(590,270,606,286,'+'); Mnu.ButtonOut(590,290,606,306,'+');
    Mnu.ButtonOut(590,310,606,326,'+'); Mnu.ButtonOut(590,402,606,418,'+');
    Mnu.ButtonOut(610,402,626,418,'-'); Mnu.ButtonOut(610,270,626,286,'-');
    Mnu.ButtonOut(610,290,626,306,'-'); Mnu.ButtonOut(610,310,626,326,'-');
    Mnu.ButtonOut(8,452,68,471,'Fertig');
    Mnu.ButtonOut(72,452,162,471,'Speichern');

```

```

Mnu.ButtonOut(166,452,228,471,'Laden');
Mnu.ButtonOut(232,452,272,471,'Neu');
Mnu.ButtonOut(276,452,353,471,'Loeschen');
Mnu.ButtonOut(357,452,416,471,'Start');
ColorButtons(color,color2);
RadioButtons(koordtyp);
M.Show(False);
End;
{-----Maximale Anzahl der Ebenen schreiben-----}
Procedure WriteMaxCellNumber(typ:Byte);
Var st: String;
    zmax: Byte;
Begin
    SetColor(14);
    SetFillStyle(1,14);
    SolidBar(545,406,560,414);
    Case typ Of
        1:zmax:=ZellenZ;
        2:zmax:=ZellenY;
        3:zmax:=ZellenX;
    End;
    str(zmax:2,st);
    SetText(545,406,st,15,0);
End;
{-----Maximale Zellenanzahl von x-, y- und z-Richtung schreiben-----}
Procedure WriteCellNumber(typ:Byte);
Var st: String;
Begin
    SetColor(14);
    SetFillStyle(1,14);
    SolidBar(560,274,576,322);
    str(ZellenX:2,st);
    SetText(560,274,st,15,0);
    str(ZellenY:2,st);
    SetText(560,294,st,15,0);
    str(ZellenZ:2,st);
    SetText(560,314,st,15,0);
    WriteMaxCellNumber(typ);
End;
{-----Aktuelle Position der Ebene schreiben-----}
Procedure WriteLevelNumber;
Var st: String;
Begin
    SetColor(14);
    SetFillStyle(1,14);
    SolidBar(490,406,510,414);
    Case koordtyp Of
        1:If level>ZellenZ Then level:=ZellenZ;
        2:If level>ZellenY Then level:=ZellenY;
        3:If level>ZellenX Then level:=ZellenX;
    End;
    str(level:2,st);
    SetText(490,406,st,15,0);

```

```

End;
{-----Darstellen der "Radiobuttons"-----}
Procedure RadioButtons(typ:Byte);
Begin
  M.Show(False);
  Mnu.ButtonIn(610,336,626,352,"");
  Mnu.ButtonIn(610,356,626,372,"");
  Mnu.ButtonIn(610,376,626,392,"");
  SetFillStyle(1,0);
  Case typ Of
    1:Begin
      FillEllipse(618,344,4,4);
      If level>ZellenZ Then Begin
        level:=ZellenZ;
        WriteLevelNumber;
      End;
    End;
    2:Begin
      FillEllipse(618,364,4,4);
      If level>ZellenY Then Begin
        level:=ZellenY;
        WriteLevelNumber;
      End;
    End;
    3:Begin
      FillEllipse(618,384,4,4);
      If level>ZellenX Then Begin
        level:=ZellenX;
        WriteLevelNumber;
      End;
    End;
  End;
  WriteMaxCellnumber(typ);
  M.Show(True);
End;
{-----Variablen für dreidimensionale Darstellung sichern-----}
Procedure SaveVar;
Begin
  oldux:=ux; olduy:=uy;
  oldkippw:=KippWinkel;
  olddrehw:=DrehWinkel;
  oldZoom_Faktor:=Zoom_Faktor;
  oldDistanz:=Distanz;
  oldsp:=sp;
  oldRahmenChk:=RahmenChk;
End;
{-----Den Variablen neue Werte übergeben-----}
Procedure SetVar;
Begin
  ux:=535; uy:=98;
  sp:=1;
  KippWinkel:=20;
  DrehWinkel:=-30;

```

```

Zoom_Faktor:=65;
Distanz:=400;
RahmenChk:=False;
End;
{-----Den Variablen die zuvor gesicherten Werte übergeben-----}
Procedure RestoreVar;
Begin
ux:=oldux; uy:=olduy;
KippWinkel:=oldkippw;
DrehWinkel:=olddrehw;
Zoom_Faktor:=oldZoom_Faktor;
Distanz:=oldDistanz;
sp:=oldsp;
RahmenChk:=oldRahmenChk;
End;
{-----Raster im Bereich der zweidimensionalen Darstellung zeichnen-----}
Procedure DrawLines(typ:Byte);
Var x,y,z: Integer;
    st: String;
Begin
SetColor(14);
SetFillStyle(1,14);
SolidBar(8,33,408,433);
ab:=400 div GetBiggestCell(ZellenX,ZellenY,ZellenZ);
SetColor(13);
If typ=1 Then Begin
    For x:= 0 To ZellenX Do
        Line(8,33+x*ab,8+ab*ZellenY,33+x*ab);
    For y:= 0 To ZellenY Do
        Line(8+y*ab,33,8+y*ab,33+ab*ZellenX);
    End;
If typ=2 Then Begin
    For x:= 0 To ZellenX Do
        Line(8+x*ab,33,8+x*ab,33+ab*ZellenZ);
    For z:= 0 To ZellenZ Do
        Line(8,33+z*ab,8+ab*ZellenX,33+z*ab);
    End;
If typ=3 Then Begin
    For y:= 0 To ZellenY Do
        Line(8+y*ab,33,8+y*ab,33+ab*ZellenZ);
    For z:= 0 to ZellenZ Do
        Line(8,33+z*ab,8+ab*ZellenY,33+z*ab);
    End;
End;
End;
{-----Zweidimensionale Zellen zeichnen-----}
Procedure SetzeZelle(typ,col1,col2:Byte;mx,my,mt:Integer);
Var a,b,c1,c2,c3,x2,y2: Integer;
    cc1,cc2,cc3: Integer;
    x,y,oldx,oldy,xm,ym: Integer;
    mtyp: Boolean;

{-----Position der Maus schreiben-----}
Procedure WriteMousePos(x,y:Integer);

```

```

Var stx,sty: String;
Begin
  str(x:2,sty);
  str(y:2,sty);
  setcolor(15);
  SetFillStyle(1,14);
  M.Show(False);
  SolidBar(360,11,376,19);
  SolidBar(400,11,416,19);
  SetText(360,11,sty,15,0);
  SetText(400,11,sty,15,0);
  M.Show(True);
End;

Begin
Case col1 Of
  0:Begin c1:=14;c2:=14;c3:=14; End;
  1:Begin c1:=1;c2:=2;c3:=3; End;
  2:Begin c1:=4;c2:=5;c3:=6; End;
  3:Begin c1:=7;c2:=8;c3:=9; End;
End;
Case col2 Of
  0:Begin cc1:=14;cc2:=14;cc3:=14; End;
  1:Begin cc1:=1;cc2:=2;cc3:=3; End;
  2:Begin cc1:=4;cc2:=5;cc3:=6; End;
  3:Begin cc1:=7;cc2:=8;cc3:=9; End;
End;
Case typ Of
  1:Begin x2:=7+ab*ZellenY;y2:=32+ab*ZellenX; End;
  2:Begin x2:=7+ab*ZellenX;y2:=32+ab*ZellenZ; End;
  3:Begin x2:=7+ab*ZellenY;y2:=32+ab*ZellenZ; End;
End;
If M.Innen(8,33,x2,y2,mx,my) Then Begin
  M.SetKreuz;
  Repeat
    M.Status(mx,my,mt);
    xm:=(mx-8) div ab; ym:=(my-33) div ab;
    x:=1+xm; y:=1+ym;
    If (x<>oldx) Or (y<>oldy) Then Begin
      oldx:=x; oldy:=y;
      WriteMousePos(x,y);
    End;
  If (mt=Linketaste) And M.Innen(8,33,x2,y2,mx,my) Then Begin
    If ((typ=1) And (Feld[1+ym,1+xm,level]<>color)) Or
      ((typ=2) And (Feld[ZellenX-xm,level,ZellenZ-ym]<>color)) Or
      ((typ=3) And (Feld[level,1+xm,ZellenZ-ym]<>color)) Then Begin
      M.Show(False);
      Mnu.Win(9+xm*ab,34+ym*ab,7+ab+xm*ab,32+ab+ym*ab,c3,c2,c1);
      Case koordtyp Of
        1:Feld[1+ym,1+xm,level]:=color;
        2:Feld[ZellenX-xm,level,ZellenZ-ym]:=color;
        3:Feld[level,1+xm,ZellenZ-ym]:=color;
      End;
    End;
  End;

```

```

    M.Show(True);
  End;
End;
If (mt=RechteTaste) And M.Innen(8,33,x2,y2,mx,my) Then Begin
  If ((typ=1) And (Feld[1+ym,1+xm,level]<>color2)) Or
    ((typ=2) And (Feld[ZellenX-xm,level,ZellenZ-ym]<>color2)) Or
    ((typ=3) and (Feld[level,1+xm,ZellenZ-ym]<>color2)) Then Begin
    M.Show(False);
    Mnu.Win(9+xm*ab,34+ym*ab,7+ab+xm*ab,32+ab+ym*ab,cc3,cc2,cc1);
    Case koordtyp Of
      1:Feld[1+ym,1+xm,level]:=color2;
      2:Feld[ZellenX-xm,level,ZellenZ-ym]:=color2;
      3:Feld[level,1+xm,ZellenZ-ym]:=color2;
    End;
    M.Show(True);
  End;
End;
Until M.Aussen(8,33,x2,y2,mx,my);
NeuFeld:=Feld;
SetColor(15);
SetFillStyle(1,14);
M.Show(False);
SolidBar(360,11,376,19); SolidBar(400,11,416,19);
M.Show(True);
End Else M.SetHand;
End;
{-----Zellen im zweidimensionalen Bereich zeichnen-----}
Procedure DrawField(typ,level:Byte);
Var x,y,z,vx,vx2,vy,vy2: Integer;
Begin
  If typ=1 Then Begin
    vx:=34; vx2:=32+ab;
    For x:= 1 To ZellenX Do Begin
      vy:=9; vy2:=7+ab;
      For y:= 1 to ZellenY Do Begin
        If Feld[x,y,level]=1 Then Mnu.Win(vy,vx,vy2,vx2,3,2,1)
          Else If Feld[x,y,level]=2 Then Mnu.Win(vy,vx,vy2,vx2,6,5,4)
            Else If Feld[x,y,level]=3 Then Mnu.Win(vy,vx,vy2,vx2,9,8,7);
        Inc(vy,ab); Inc(vy2,ab);
      End;
      Inc(vx,ab); Inc(vx2,ab);
    End;
  End;
End;
If typ=2 Then Begin
  vx:=9; vx2:=7+ab;
  For x:= ZellenX DownTo 1 Do Begin
    vy:=34; vy2:=32+ab;
    For z:= ZellenZ DownTo 1 Do Begin
      If Feld[x,level,z]=1 Then Mnu.Win(vx,vy,vx2,vy2,3,2,1)
        Else If Feld[x,level,z]=2 Then Mnu.Win(vx,vy,vx2,vy2,6,5,4)
          Else If Feld[x,level,z]=3 Then Mnu.Win(vx,vy,vx2,vy2,9,8,7);
      Inc(vy,ab); Inc(vy2,ab);
    End;
  End;
End;

```

```

    Inc(vx,ab); Inc(vx2,ab);
End;
End;
If typ=3 Then Begin
    vx:=9; vx2:=7+ab;
    For y:= 1 To ZellenY Do Begin
        vy:=34; vy2:=32+ab;
        For z:= ZellenZ DownTo 1 Do Begin
            If Feld[level,y,z]=1 Then Mnu.Win(vx,vy,vx2,vy2,3,2,1)
            Else If Feld[level,y,z]=2 Then Mnu.Win(vx,vy,vx2,vy2,6,5,4)
            Else If Feld[level,y,z]=3 Then Mnu.Win(vx,vy,vx2,vy2,9,8,7);
            Inc(vy,ab); Inc(vy2,ab);
        End;
        Inc(vx,ab); Inc(vx2,ab);
    End;
End;
End;
{-----Raster und Zellen zeichnen-----}
Procedure Draw2D(typ,level:Byte);
Begin
    M.Show(False);
    DrawLines(typ);
    DrawField(typ,level);
    M.Show(True);
End;
{-----Start der zweidimensionale Simulation-----}
Procedure Start2D(typ,level:Byte);
Var DrawStop: Boolean;

{-----Mausabfrage für Abbruch-Status-----}
Procedure MausAbfrage;
Begin
    M.Status(mx,my,mt);
    If mt=linketaste Then Begin
        DrawStop:=not DrawStop;
        If DrawStop Then MessageWindow('ABBRUCH','AKTIVIERT',keinetaste)
        Else MessageWindow('ABBRUCH','DEAKTIVIERT',keinetaste);
    End;
End;
{-----Zustände der Zellen einlesen-----}
Procedure Zellenlesen2D;
Var i,j,k: Byte;
Begin
    For i:= 1 To ZellenZ Do
        For j:= 1 To ZellenY Do
            For k:= 1 To ZellenX Do Begin
                Zelle_lesen(k,j,i);
                MausAbfrage;
            End;
        End;
    End;
End;
{-----Hauptteil der Prozedur Start2D-----}
Begin
    DrawStop:=False;

```

```

M.Show(False);
Repeat
  ZellenLesen2D;
  Feld:=NeuFeld;
  Inc(Datum);
  WriteDate(Datum);
  Draw2D(typ,level);
  M.Show(False);
Until DrawStop;
M.Show(True);
End;
{----Mausabfrage, ob "Radiobuttons" angeklickt wurden----}
Procedure GetRadioButtons(mx,my,mt:Integer; Var outvar:Byte);
Begin
  If (mt=linketaste) And M.Innen(610,336,626,352,mx,my) And (koordtyp<>1) Then Begin
    Repeat
      M.Status(mx,my,mt)
    Until mt=keinetaste;
    If M.Innen(610,336,626,352,mx,my) Then outvar:=1;
  End;
  If (mt=linketaste) And M.Innen(610,356,626,372,mx,my) And (koordtyp<>2) Then Begin
    Repeat
      M.Status(mx,my,mt)
    Until mt=keinetaste;
    If M.Innen(610,356,626,372,mx,my) Then outvar:=2;
  End;
  If (mt=linketaste) And M.Innen(610,376,626,392,mx,my) And (koordtyp<>3) Then Begin
    Repeat
      M.Status(mx,my,mt)
    Until mt=keinetaste;
    If M.Innen(610,376,626,392,mx,my) Then outvar:=3;
  End;
End;
{----Aktuelle Maustastenbelegung für Editierung zeigen----}
Procedure ColorButtons(color,color2:Byte);
Var xpos:Integer;
Begin
  M.Show(False);
  Mnu.Win(434,428,480,458,15,14,13); Mnu.Win(482,428,528,458,3,2,1);
  Mnu.Win(530,428,576,458,6,5,4); Mnu.Win(578,428,624,458,9,8,7);
  Case color Of
    0:xpos:=440;
    1:xpos:=488;
    2:xpos:=536;
    3:xpos:=584;
  End;
  SetText(xpos,440,'L',15,0);
  Case Color2 Of
    0:xpos:=466;
    1:xpos:=514;
    2:xpos:=562;
    3:xpos:=610;
  End;
End;

```



```

SetText(xpos,440,'R',15,0);
M.Show(True);
End;
{----Mausabfrage, ob Farbschaltflächen "gedrückt" wurden----}
Procedure GetColorButtons(mx,my,mt:Integer);
Begin
If M.Innen(434,428,480,458,mx,my) And ((mt=1) Or (mt=2)) Then Begin
If mt=linketaste Then color:=0 Else
If mt=rechtetaste Then color2:=0;
ColorButtons(color,color2);
Repeat
M.Status(mx,my,mt)
Until mt=keinetaste;
End Else
If M.Innen(482,428,528,458,mx,my) And ((mt=1) Or (mt=2)) Then Begin
If mt=linketaste Then color:=1 Else
If mt=rechtetaste Then color2:=1;
ColorButtons(color,color2);
Repeat
M.Status(mx,my,mt)
Until mt=keinetaste;
End Else
If M.Innen(530,428,576,458,mx,my) And ((mt=1) Or (mt=2)) Then Begin
If mt=linketaste Then color:=2 Else
If mt=rechtetaste Then color2:=2;
ColorButtons(color,color2);
Repeat
M.Status(mx,my,mt)
Until mt=keinetaste;
End Else
If M.Innen(578,428,624,458,mx,my) And ((mt=1) Or (mt=2)) Then Begin
If mt=linketaste Then color:=3 Else
If mt=rechtetaste Then color2:=3;
ColorButtons(color,color2);
Repeat
M.Status(mx,my,mt);
Until mt=keinetaste;
End;
End;
{----Löschen einer Ebene----}
Procedure ClearLevel(koordtyp,level:Byte);
Var x,y,i,j:Integer;
Begin
Case Koordtyp Of
1:Begin x:=ZellenX; y:=ZellenY; End;
2:Begin x:=ZellenX; y:=ZellenZ; End;
3:Begin x:=ZellenY; y:=ZellenZ; End;
End;
For i:=1 To x Do
For j:= 1 To y Do
Case koordtyp Of
1:Feld[i,j,level]:=0;
2:Feld[i,level,j]:=0;

```

```

    3:Feld[level,i,j]:=0;
End;
End;
{----Ständige Abfrage des Mausstatus und Ausführung der Befehle----}
Procedure EditorMausAbfrage;
Var xplus,xminus,yplus,yminus,zplus,zminus: Boolean;
    aktual1,levelplus,levelminus,EndofEditor: Boolean;
    aktual2,aktual3,load,save,neu,clear,start,code: Boolean;
    zmax,radiochk,oldzx,oldzy,oldzz: Byte;
    i: Integer;
Begin
    oldzx:=ZellenX; oldzy:=ZellenY; oldzz:=ZellenZ;
    aktual1:=False; aktual2:=False; aktual3:=False;
    xplus:=False; xminus:=False;
    yplus:=False; yminus:=False;
    zplus:=False; zminus:=False;
    levelplus:=False; levelminus:=False;
    Endofeditor:=False; load:=False; save:=False;
    neu:=False; clear:=False; start:=False;
    radiochk:=0;
Repeat
    M.Status(mx,my,mt);
    SetzeZelle(koordtyp,color,color2,mx,my,mt);
    Mnu.GetButton(590,270,606,286,linketaste,mx,my,mt,'+',xplus);
    Mnu.GetButton(590,290,606,306,linketaste,mx,my,mt,'+',yplus);
    Mnu.GetButton(590,310,606,326,linketaste,mx,my,mt,'+',zplus);
    Mnu.GetButton(610,270,626,286,linketaste,mx,my,mt,'-',xminus);
    Mnu.GetButton(610,290,626,306,linketaste,mx,my,mt,'-',yminus);
    Mnu.GetButton(610,310,626,326,linketaste,mx,my,mt,'-',zminus);
    Mnu.GetButton(590,402,606,418,linketaste,mx,my,mt,'+',levelplus);
    Mnu.GetButton(610,402,626,418,linketaste,mx,my,mt,'-',levelminus);
    Mnu.GetButton(8,452,68,471,linketaste,mx,my,mt,'Fertig',Endofeditor);
    Mnu.GetButton(72,452,162,471,linketaste,mx,my,mt,'Speichern',Save);
    Mnu.GetButton(166,452,228,471,linketaste,mx,my,mt,'Laden',load);
    Mnu.GetButton(232,452,272,471,linketaste,mx,my,mt,'Neu',neu);
    Mnu.GetButton(276,452,353,471,linketaste,mx,my,mt,'Loeschen',clear);
    Mnu.GetButton(357,452,416,471,linketaste,mx,my,mt,'Start',start);
    Mnu.GetButton(454,220,605,240,linketaste,mx,my,mt,'Aktualisieren',aktual1);
    GetRadioButtons(mx,my,mt,radiochk);
    For i:= 1 To 3 Do
        If RadioChk=i Then Begin
            If (oldzx<>ZellenX) Or (oldzy<>ZellenY) Or (oldzz<>ZellenZ) Then Begin
                oldzx:=ZellenX; oldzy:=ZellenY; oldzz:=ZellenZ;
                aktual3:=True
            End Else aktual2:=True;
            radiochk:=0; koordtyp:=i;
            RadioButtons(koordtyp);
            Draw2D(koordtyp,level);
        End;
    GetColorButtons(mx,my,mt);
    If Load Then Begin
        Load:=False;
        LadenSichernWindow(True,True,code);

```

```

If Code=False Then Begin
  Draw2D(koordtyp,level);
  WriteMaxCellNumber(koordtyp);
  WriteCellNumber(koordtyp);
  aktual1:=True;
End;
End Else
If Save Then Begin
  Save:=False;
  LadenSichernWindow(False,True,code);
End Else
If Clear Then Begin
  Clear:=False;
  ClearLevel(koordtyp,level);
  Draw2d(koordtyp,level);
  NeuFeld:=Feld;
End Else
If Neu Then Begin
  Neu:=False;
  Aktual3:=True;
  FeldReset;
  Draw2d(koordtyp,level);
End Else
If Start Then Begin
  start:=False;
  Start2D(koordtyp,level);
End Else
If aktual1 Or aktual2 Or aktual3 Then Begin
  aktual2:=False;
  Var_Ausrechnen;
  M.Show(False);
  DrawActLevel(koordtyp,level);
  If aktual3 Or aktual1 Then Begin
    aktual3:=False;
    SetColor(14);
    SetFillStyle(1,14);
    SolidBar(428,8,631,219);
    Raster(True);
    DrawActLevel(koordtyp,level);
    If aktual1 Then Begin
      aktual1:=False;
      ZellenSetzen(False);
    End;
  End;
  M.Show(True);
End Else
If levelplus Then Begin
  levelplus:=False;
  Case koordtyp Of
    1:zmax:=ZellenZ;
    2:zmax:=ZellenY;
    3:zmax:=ZellenX;
  End;

```

```

If level<zmax Then Begin
  If (oldzx<>ZellenX) Or (oldzy<>ZellenY) Or (oldzz<>ZellenZ) Then Begin
    oldzx:=ZellenX; oldzy:=ZellenY; oldzz:=ZellenZ;
    aktual3:=True
  End Else aktual2:=True;
  Inc(level);
  WriteLevelNumber;
  Draw2D(koordtyp,level);
End;
End Else
If levelminus Then Begin
  levelminus:=False;
  If level>1 Then Begin
    If (oldzx<>ZellenX) Or (oldzy<>ZellenY) Or (oldzz<>ZellenZ) Then Begin
      oldzx:=ZellenX; oldzy:=ZellenY; oldzz:=ZellenZ;
      aktual3:=True
    End Else aktual2:=True;
    Dec(level);
    WriteLevelNumber;
    Draw2D(koordtyp,level);
  End;
End Else
If xplus Then Begin
  xplus:=False;
  If ZellenX<MaxZellAnzahl_X Then Begin
    Inc(ZellenX);
    WriteCellNumber(koordtyp);
    WriteLevelNumber;
    Draw2D(koordtyp,level);
  End;
End Else
If yplus Then Begin
  yplus:=False;
  If ZellenY<MaxZellAnzahl_Y Then Begin
    Inc(ZellenY);
    WriteCellNumber(koordtyp);
    WriteLevelNumber;
    Draw2D(koordtyp,level);
  End;
End Else
If zplus Then Begin
  zplus:=False;
  If ZellenZ<MaxZellAnzahl_Z Then Begin
    Inc(ZellenZ);
    WriteCellNumber(koordtyp);
    WriteLevelNumber;
    Draw2D(koordtyp,level);
  End;
End Else
If xminus Then Begin
  xminus:=False;
  If ZellenX>4 Then Begin
    Dec(ZellenX);

```

```

    WriteCellNumber(koordtyp);
    WriteLevelNumber;
    Draw2D(koordtyp,level);
End;
End Else
If yminus Then Begin
yminus:=False;
If ZellenY>4 Then Begin
    Dec(ZellenY);
    WriteCellNumber(koordtyp);
    WriteLevelNumber;
    Draw2D(koordtyp,level);
End;
End Else
If zminus Then Begin
zminus:=False;
If ZellenZ>4 Then Begin
    Dec(ZellenZ);
    WriteCellNumber(koordtyp);
    WriteLevelnumber;
    Draw2D(koordtyp,level);
End;
End;
Until EndofEditor=True;
End;
{-----Hauptteil der Prozedur Bearbeiten-----}
Begin
M.Show(False);
GetScreen;
Init;
SaveVar;
SetVar;
Var_Ausrechnen;
M.Show(False);
Raster(True);
DrawActLevel(koordtyp,level);
WriteCellNumber(koordtyp);
WriteLevelNumber;
ZellenSetzen(False);
Draw2d(koordtyp,level);
M.Show(True);
EditorMausAbfrage;
M.Show(False);
PutScreen;
RestoreVar;
Var_Ausrechnen;
RasterZeichnen(True);
M.Show(False);
WriteCellStatus(False);
M.Show(True);
End;
{-----Ändern des Zellabstands-----}
Procedure ChangeCellSpace;

```

```

Var size: Word;
  p: Pointer;
  st: String;
  up,down,Ende: Boolean;
Begin
  M.Show(False);
  up:=False; down:=False; Ende:=False;
  size:=ImageSize(250,200,390,320);
  GetMem(p,size);
  GetImage(250,200,390,320,p^);
  Mnu.Win(250,200,390,320,15,14,13);
  Mnu.Win(254,204,386,219,13,14,15);
  Mnu.Win(254,223,386,316,13,14,15);
  SetText(250+(140-textwidth('Zellabstand')) div 2,208,'Zellabstand',15,0);
  SetText(330,230+(45-textheight('Pixel')) div 2,'Pixel',0,15);
  Mnu.ButtonOut(260,230,280,250,#30);
  Mnu.ButtonOut(260,255,280,275,#31);
  Mnu.ButtonOut(270,290,370,310,'Ok');
  str(sp,st);
  Mnu.ButtonIn(300,241,320,263,st);
  M.MWindow(250,200,390,320);
  M.Show(True);
  Repeat
    M.Status(mx,my,mt);
    Mnu.GetButton(260,230,280,250,linketaste,mx,my,mt,#30,up);
    Mnu.GetButton(260,255,280,275,linketaste,mx,my,mt,#31,down);
    Mnu.GetButton(270,290,370,310,linketaste,mx,my,mt,'Ok',Ende);
    If up Then Begin
      up:=False;
      If sp<6 Then Begin
        Inc(sp);
        str(sp,st);
        Mnu.ButtonIn(300,241,320,263,st);
      End;
    End Else
    If down Then Begin
      down:=False;
      If sp>0 Then Begin
        Dec(sp);
        str(sp,st);
        Mnu.ButtonIn(300,241,320,263,st);
      End;
    End;
  Until Ende;
  M.Show(False);
  M.MWindow(0,0,maxX,maxY);
  PutImage(250,200,p^,0);
  FreeMem(p,size);
  M.Show(True);
End;
{----Ändern der Zellnachbarkonfiguration----}
Procedure SetNeighBors;
Var size: Word;

```

```

p: Pointer;
oldux,olduy,olddrehw,oldkippw,oldDistanz,oldZoom_Faktor: Integer;
oldsp: Shortint;
seq,oldZellenX,oldZellenY,oldZellenZ: Byte;
oldRahmenChk,Ende,standard,rb,left,right: Boolean;

```

```
{----Werte der Variablen für dreidimensionale Darstellung sichern----}
```

```
Procedure SaveVar;
```

```
Begin
```

```

oldux:=ux; olduy:=uy;
oldZellenX:=ZellenX; oldZellenY:=ZellenY; oldZellenZ:=ZellenZ;
oldkippw:=KippWinkel; olddrehw:=DrehWinkel;
oldZoom_Faktor:=Zoom_Faktor; oldDistanz:=Distanz;
oldsp:=sp; oldRahmenChk:=RahmenChk; NeuFeld:=Feld;

```

```
End;
```

```
{----Den Variablen neue Werte übergeben----}
```

```
Procedure SetVar;
```

```
Begin
```

```

ux:=230; uy:=208; sp:=4; KippWinkel:=30; DrehWinkel:=-30;
Zoom_Faktor:=44; Distanz:=230; RahmenChk:=True;
ZellenX:=3; ZellenY:=3; ZellenZ:=3;

```

```
End;
```

```
{----Gespeicherte Werte den Variablen übergeben----}
```

```
Procedure RestoreVar;
```

```
Begin
```

```

ux:=oldux; uy:=olduy; KippWinkel:=oldkippw; DrehWinkel:=olddrehw;
Zoom_Faktor:=oldZoom_Faktor; Distanz:=oldDistanz;
sp:=oldsp; RahmenChk:=oldRahmenChk; Feld:=NeuFeld;
ZellenX:=oldZellenX; ZellenZ:=oldZellenZ; ZellenY:=oldZellenY;

```

```
End;
```

```
{----Aktivierte Kontrollkästchen zeigen----}
```

```
Procedure PlacePoints;
```

```
Begin
```

```

SetColor(0);
SetFillStyle(1,0);
If nb[seq,1] Then fillellipse(310,165,5,5);
If nb[seq,2] Then fillellipse(310,190,5,5);
If nb[seq,3] Then fillellipse(310,215,5,5);
If nb[seq,4] Then fillellipse(310,240,5,5);
SetColor(14);
SetFillStyle(1,14);
If nb[seq,1]=False Then fillellipse(310,165,5,5);
If nb[seq,2]=False Then fillellipse(310,190,5,5);
If nb[seq,3]=False Then fillellipse(310,215,5,5);
If nb[seq,4]=False Then fillellipse(310,240,5,5);

```

```
End;
```

```
{----Anzahl der aktiven Nachbarn schreiben----}
```

```
Procedure ActiveNb;
```

```
Var i,j,k,z: Integer;
```

```
st: String;
```

```
Begin
```

```

z:=0;
For i:=1 to 3 do

```

```

For j:= 1 to 3 do
  For k:= 1 to 3 do
    If Feld[i,j,k]<>0 Then Inc(z);
  SetFillStyle(1,14);
  SetColor(14);
  SolidBar(440,265,456,273);
  str(z:2,st);
  SetText(440,265,st,15,0);
End;
{----Flächennachbarn aktivieren----}
Procedure SetFINb;
Begin
  If nb[seq,1] Then Begin
    Feld[2,2,1]:=1;
    Feld[2,1,2]:=1;
    Feld[1,2,2]:=1;
    Feld[2,3,2]:=1;
    Feld[3,2,2]:=1;
    Feld[2,2,3]:=1;
  End Else Begin
    Feld[2,2,1]:=0;
    Feld[2,1,2]:=0;
    Feld[1,2,2]:=0;
    Feld[2,3,2]:=0;
    Feld[3,2,2]:=0;
    Feld[2,2,3]:=0;
  End;
End;
{----Kantennachbarn aktivieren----}
Procedure SetKtNb;
Begin
  If nb[seq,2] Then Begin
    Feld[2,1,1]:=2;
    Feld[1,2,1]:=2;
    Feld[3,2,1]:=2;
    Feld[2,3,1]:=2;
    Feld[1,1,2]:=2;
    Feld[3,1,2]:=2;
    Feld[1,3,2]:=2;
    Feld[3,3,2]:=2;
    Feld[2,1,3]:=2;
    Feld[1,2,3]:=2;
    Feld[3,2,3]:=2;
    Feld[2,3,3]:=2;
  End Else Begin
    Feld[2,1,1]:=0;
    Feld[1,2,1]:=0;
    Feld[3,2,1]:=0;
    Feld[2,3,1]:=0;
    Feld[1,1,2]:=0;
    Feld[3,1,2]:=0;
    Feld[1,3,2]:=0;
    Feld[3,3,2]:=0;
  End;
End;

```



```

    Feld[2,1,3]:=0;
    Feld[1,2,3]:=0;
    Feld[3,2,3]:=0;
    Feld[2,3,3]:=0;
End;
End;
{-----Ecknachbarn aktivieren-----}
Procedure SetEkNb;
Begin
    If nb[seq,3] Then Begin
        Feld[1,1,1]:=3;
        Feld[1,3,1]:=3;
        Feld[3,1,1]:=3;
        Feld[3,3,1]:=3;
        Feld[1,1,3]:=3;
        Feld[1,3,3]:=3;
        Feld[3,1,3]:=3;
        Feld[3,3,3]:=3;
    End Else Begin
        Feld[1,1,1]:=0;
        Feld[1,3,1]:=0;
        Feld[3,1,1]:=0;
        Feld[3,3,1]:=0;
        Feld[1,1,3]:=0;
        Feld[1,3,3]:=0;
        Feld[3,1,3]:=0;
        Feld[3,3,3]:=0;
    End;
End;
{-----Nachbarn aktivieren-----}
Procedure SetNB;
Begin
    SetFINb;
    SetKtNb;
    SetEkNb;
    If nb[seq,4] Then Feld[2,2,2]:=3
        Else Feld[2,2,2]:=0;
End;
{-----Dreidimensionale Darstellung löschen und neu aufbauen-----}
Procedure ClearAndNew;
Begin
    M.Show(False);
    SetFillStyle(1,14);
    SetColor(14);
    SolidBar(157,150,298,280);
    Raster(True);
    SetNb;
    ActiveNb;
    ZellenSetzen(False);
    M.Show(True);
End;
{-----Reihenfolge-Schaltflächen zeichnen-----}
Procedure PlaceSeqButtons;

```

```

Begin
  If SeqAnz>=1 Then
    If seq=1 Then Mnu.ButtonIn(300,305,330,325,'1')
      Else Mnu.ButtonOut(300,305,330,325,'1');
  If SeqAnz>=2 Then
    If seq=2 Then Mnu.ButtonIn(340,305,370,325,'2')
      Else Mnu.ButtonOut(340,305,370,325,'2')
  Else SolidBar(340,305,370,325);
  If SeqAnz>=3 Then
    If seq=3 Then Mnu.ButtonIn(380,305,410,325,'3')
      Else Mnu.ButtonOut(380,305,410,325,'3')
  Else SolidBar(380,305,410,325);
  If SeqAnz>=4 Then
    If seq=4 Then Mnu.ButtonIn(420,305,450,325,'4')
      Else Mnu.ButtonOut(420,305,450,325,'4')
  Else SolidBar(420,305,450,325);
End;
{----Mausabfrage für Elemente mit Kontrollkästchen----}
Procedure GetRadioButton(x1,y1,x2,y2:Integer;mxpos,mypos,taste:Integer;Var chk1,chk2:Boolean);
Var x,y,t,xpos,ypos:Integer;
Begin
  If (taste=linketaste) And M.Innen(x1,y1,x2,y2,mxpos,mypos) Then Begin
    xpos:=x1+(x2-x1) div 2;
    ypos:=y1+(y2-y1) div 2;
    M.Show(False);
    If chk1 Then Begin
      SetFillStyle(1,14);
      SetColor(14);
      FillEllipse(xpos,ypos,xpos-5-x1,ypos-5-y1);
    End Else Begin
      SetFillStyle(1,0);
      SetColor(0);
      FillEllipse(xpos,ypos,xpos-5-x1,ypos-5-y1);
    End;
    M.Show(True);
    Repeat
      M.Status(x,y,t);
    Until t=keinetaste;
    chk1:=not chk1;
    chk2:=True;
  End;
End;
{----Mausabfrage für Reihenfolge-Schaltflächen----}
Procedure GetSeqButton(x1,y1,x2,y2,xpos,ypos,taste:Integer;sq:Byte;Var seq:Byte);
Var x,y,t: Integer;
    st: String;
Begin
  If (taste=linketaste) And M.Innen(x1,y1,x2,y2,xpos,ypos) And (seq<>sq) Then Begin
    seq:=sq;
    str(seq,st);
    M.Show(False);
    PlaceSeqButtons;
    Mnu.ButtonIn(x1,y1,x2,y2,st);
  End;
End;

```

```

    PlacePoints;
    ClearAndNew;
    M.Show(True);
    Repeat
        M.Status(x,y,t);
    Until t=keinetaste;
End;
End;
{-----Hauptteil der Prozedur Setneighbors-----}
Begin
    SaveVar;
    SetVar;
    Var_Ausrechnen;
    seq:=1;
    SetNb;
    rb:=False;
    Ende:=False; left:=False; right:=False; standard:=False;
    M.Show(False);
    size:=ImageSize(150,120,490,380);
    GetMem(p,size);
    GetImage(150,120,490,380,p^);
    Mnu.Win(150,120,490,380,15,14,13); Mnu.Win(154,124,486,139,13,14,15);
    Mnu.Win(154,143,486,292,13,14,15); Mnu.Win(154,296,486,334,13,14,15);
    Mnu.ButtonOut(180,345,300,369,'STANDARD');
    Mnu.ButtonOut(340,345,460,369,'OK');
    Mnu.ButtonIn(300,155,320,175,""); Mnu.ButtonIn(300,180,320,200,"");
    Mnu.ButtonIn(300,205,320,225,""); Mnu.ButtonIn(300,230,320,250,"");
    PlaceSeqButtons;
    Mnu.ButtonIn(300,305,330,325,'1');
    Mnu.ButtonOut(170,315,210,327,#17);
    Mnu.ButtonOut(222,315,262,327,#16);
    SetText(325,161,'Flächennachbarn',0,15);
    SetText(325,186,'Kantennachbarn',0,15);
    SetText(325,211,'Ecknachbarn',0,15);
    SetText(325,236,'Zelle im Zentrum',0,15);
    SetText(300,265,'Aktive Nachbarn:',0,15);
    SetText(170,303,'Reihenfolge:',0,15);
    PlacePoints;
    ActiveNb;
    SetText(150+(340-textwidth('Zellnachbarn')) div 2,128,'Zellnachbarn',15,0);
    Raster(True);
    Zellensetzen(False);
    M.MWindow(150,120,490,380);
    M.Show(True);
    Repeat
        M.Status(mx,my,mt);
        GetRadioButton(300,155,320,175,mx,my,mt,nb[seq,1],rb);
        GetRadioButton(300,180,320,200,mx,my,mt,nb[seq,2],rb);
        GetRadioButton(300,205,320,225,mx,my,mt,nb[seq,3],rb);
        GetRadioButton(300,230,320,250,mx,my,mt,nb[seq,4],rb);
        Mnu.GetButton(180,345,300,369,linketaste,mx,my,mt,'STANDARD',standard);
        Mnu.GetButton(340,345,460,369,linketaste,mx,my,mt,'OK',Ende);
        Mnu.GetButton(170,315,210,327,linketaste,mx,my,mt,#17,left);

```

```

Mnu.GetButton(222,315,262,327,linketaste,mx,my,mt,#16,right);
If seqanz>=1 Then GetSeqButton(300,305,330,325,mx,my,mt,1,seq);
If seqanz>=2 Then GetSeqButton(340,305,370,325,mx,my,mt,2,seq);
If seqanz>=3 Then GetSeqButton(380,305,410,325,mx,my,mt,3,seq);
If seqanz>=4 Then GetSeqButton(420,305,450,325,mx,my,mt,4,seq);
If left Then Begin
  left:=False;
  If seqanz>1 Then Begin
    Dec(SeqAnz);
    If seqanz<seq Then Begin
      seq:=seqanz;
      PlacePoints;
      ClearAndNew;
    End;
    M.Show(False);
    PlaceSeqButtons;
    M.Show(True);
  End;
End Else
If right Then Begin
  right:=False;
  If seqanz<4 Then Begin
    Inc(SeqAnz);
    M.Show(False);
    PlaceSeqButtons;
    M.Show(True);
  End;
End else
If rb Then Begin
  rb:=False;
  ClearAndNew;
End else
If standard Then Begin
  standard:=False;
  NeighborInit;
  seq:=1;
  PlaceSeqButtons;
  PlacePoints;
  ClearAndNew;
End;
Until Ende=True;
M.Show(False);
M.MWindow(0,0,maxX,maxY);
PutImage(150,120,p^,0);
FreeMem(p,size);
RestoreVar;
Var_Ausrechnen;
M.Show(True);
End;
{----Informationsfenster darstellen----}
Procedure InfoBox(x1,y1,x2,y2:Integer);
Const title='Info über Life 3D...';
  txt1='LIFE 3D';

```

```

    txt2='Version 1.1A';
    txt3='Copyright (c) 1994/95 By';
    txt4='Roland Wohlfahrt';
Var size: Word;
    p: Pointer;
    ok: Boolean;
Begin
    M.Show(False);
    M.MWindow(x1,y1,x2,y2);
    size:=ImageSize(x1,y1,x2,x2);
    GetMem(p,size);
    GetImage(x1,y1,x2,y2,p^);
    Mnu.Win(x1,y1,x2,y2,15,14,13);
    Mnu.Win(x1+4,y1+4,x2-4,y1+20,13,14,15);
    Mnu.Win(x1+4,y1+24,x2-4,y2-4,13,14,15);
    Mnu.ButtonOut((x1+x2) div 2-20,y2-32,(x1+x2) div 2+20,y2-12,'Ok');
    SetText(x1+(x2-x1-textwidth(title)) div 2,y1+8,title,6,0);
    SetText(x1+(x2-x1-textwidth(txt1)) div 2,y1+40,txt1,15,0);
    SetText(x1+(x2-x1-textwidth(txt2)) div 2,y1+60,txt2,15,0);
    SetText(x1+(x2-x1-textwidth(txt3)) div 2,y1+80,txt3,15,0);
    SetText(x1+(x2-x1-textwidth(txt4)) div 2,y1+100,txt4,15,0);
    ok:=False;
    M.Show(True);
    Repeat
        M.Status(mx,my,mt);
        Mnu.GetButton((x1+x2) div 2-20,y2-32,(x1+x2) div 2+20,y2-12,
            linketaste,mx,my,mt,'Ok',ok);
    Until Ok;
    M.Show(False);
    PutImage(x1,y1,p^,0);
    FreeMem(p,size);
    M.MWindow(0,0,maxx,maxy);
    M.Show(True);
End;
{-----Abfrage des Mausstatus im Hauptbildschirm-----}
Procedure MenuAbfrage;
Const Otext:Array[0..5] Of String=('3D-Raster      AUS',
    'Zellrahmen      AUS',
    'Zelluniversum    UNENDLICH',
    'Zellenabstand einstellen...',
    'Nachbarn festlegen...  ',
    'Datum auf Null setzen  ');
    Dtext:Array[0..5] Of String=('Neu          ',
    'Bild laden...        ',
    'Bild speichern...    ',
    'Konfiguration laden... ',
    'Konfiguration speichern...',
    'Programm beenden     ');
Var jump,bool,draw,optio,datei,bearb,help,beding : Boolean;
    up,Down,left,right,plus,minus,standard : Boolean;
    fullpic,start,distplus,distminus,full,code : Boolean;
    Quest: Boolean;

```

```

    ov: Byte;
Begin
  FullPic:=False; Draw:=False; Start:=False; Optio:=False; Datei:=False;
  Bearb:=False; Beding:=False; Help:=False; Bool:=False; Jump:=False;
  Full:=False; Standard:=False; DistPlus:=False; DistMinus:=False;
  Plus:=False; Minus:=False; Up:=False; Down:=False; Right:=False;
  Left:=False;
Repeat
  M.Status(mx,my,mt);
  Mnu.GetButton(14,445,86,465,linketaste,mx,my,mt,'Start',start);
  Mnu.GetButton(14,420,86,440,linketaste,mx,my,mt,'Sprung',Jump);
  Mnu.GetButton(14,395,86,415,linketaste,mx,my,mt,'Zeichnen',draw);
  Mnu.GetSwitch(14,370,86,390,linketaste,mx,my,mt,'Vollbild','Vollbild',Full,fullpic);
  Mnu.GetButton(10,4,90,26,linketaste,mx,my,mt,mtext[1],Datei);
  Mnu.GetButton(100,4,204,26,linketaste,mx,my,mt,mtext[2],Optio);
  Mnu.GetButton(214,4,302,26,linketaste,mx,my,mt,mtext[3],Bearb);
  Mnu.GetButton(312,4,456,26,linketaste,mx,my,mt,mtext[4],Beding);
  Mnu.GetButton(466,4,538,26,linketaste,mx,my,mt,mtext[5],Bool);
  Mnu.GetButton(28,322,48,342,linketaste,mx,my,mt,'+',distplus);
  Mnu.GetButton(52,322,72,342,linketaste,mx,my,mt,'-',distminus);
  Mnu.GetButton(40,158,60,178,linketaste,mx,my,mt,#30,up);
  Mnu.GetButton(40,202,60,222,linketaste,mx,my,mt,#31,Down);
  Mnu.GetButton(18,180,38,200,linketaste,mx,my,mt,#17,left);
  Mnu.GetButton(62,180,82,200,linketaste,mx,my,mt,#16,right);
  Mnu.GetButton(28,262,48,282,linketaste,mx,my,mt,'+',plus);
  Mnu.GetButton(52,262,72,282,linketaste,mx,my,mt,'-',minus);
  Mnu.GetButton(40,180,60,200,linketaste,mx,my,mt,'S',standard);
  If FullPic Then Begin
    FullPic:=False;
    If Full Then
      FullScreenChk:=True
    Else FullScreenChk:=False;
  End Else
  If Start Then Begin
    Start:=False;
    StartDraw_Cells;
  End Else
  If Draw Then Begin
    Draw:=False;
    M.Show(False);
    Draw_Cells;
    M.Show(True);
  End Else
  If Datei Then Begin
    Datei:=False;
    ov:=0;
    Mnu.Window(10,32,240,162,dtext,ov);
    If ov=1 Then Begin
      FeldReset;
      ResetVar;
      M.Show(False);
      Datum:=0;
      ZellenX:=15;
    End
  End
End Repeat

```

```

ZellenY:=15;
ZellenZ:=15;
WriteCellStatus(False);
M.Show(True);
DrawScreen;
ClearViewPort;
FullScreen;
RasterZeichnen(True);
End Else
If ov=2 Then Begin
  LadenSichernWindow(True,True,code);
  M.Show(False);
  If Code=False Then Begin
    WriteCellStatus(FullScreenChk);
    Draw_Cells;
  End;
  M.Show(True);
End Else
If ov=3 Then LadenSichernWindow(False,True,code) Else
If ov=4 Then LadenSichernWindow(True,False,code) Else
If ov=5 Then LadenSichernWindow(False,False,code) Else
If ov=6 Then EndOfProg:=True;
End Else
If Bearb Then Begin
  Bearb:=False;
  Bearbeiten;
End Else
If Beding Then Begin
  Beding:=False;
  Bedingungen;
End Else
If Optio Then Begin
  Optio:=False;
  ov:=0;
  Delete(otext[1],25,3);
  Delete(otext[0],25,3);
  Delete(otext[2],19,9);
  If RahmenChk Then insert('EIN',otext[1],25)
    Else insert('AUS',otext[1],25);
  If RasterChk Then insert('EIN',otext[0],25)
    Else insert('AUS',otext[0],25);
  If FeldTypChk Then insert('UNENDLICH',otext[2],19)
    Else insert(' ENDLICH',otext[2],19);
  Mnu.Window(100,32,338,162,otext,ov);
  If ov=1 Then RasterChk:=Not RasterChk Else
  If ov=2 Then RahmenChk:=Not RahmenChk Else
  If ov=3 Then FeldTypChk:=Not FeldTypChk Else
  If ov=4 Then ChangeCellSpace Else
  If ov=5 Then SetNeighbors Else
  If ov=6 Then Begin
    Datum:=0;
    M.Show(False);
    WriteCellStatus(False);

```

```

    M.Show(True);
End;
End Else
If Bool Then Begin
    Bool:=False;
    InfoBox(200,160,440,320);
End Else
If Jump Then Begin
    Jump:=False;
    Sprung;
End Else
If Plus Then Begin
    Plus:=False;
    If Zoom_Faktor<300 Then Begin
        Inc(Zoom_Faktor,10);
        RasterZeichnen(True);
    End;
End Else
If Minus Then Begin
    Minus:=False;
    If Zoom_Faktor>50 Then Begin
        Dec(Zoom_Faktor,10);
        RasterZeichnen(True);
    End;
End Else
If DistPlus Then Begin
    DistPlus:=False;
    If Distanz<1500 Then Begin
        Inc(Distanz,100);
        RasterZeichnen(True);
    End;
End Else
If DistMinus Then Begin
    DistMinus:=False;
    If Distanz>600 Then Begin
        Dec(Distanz,100);
        RasterZeichnen(True);
    End;
End Else
If Standard Then Begin
    Standard:=False;
    Zoom_Faktor:=150;
    KippWinkel:=30;
    DrehWinkel:=-10;
    RasterZeichnen(True);
End Else
If Left Then Begin
    Left:=False;
    If DrehWinkel<40 Then Begin
        Inc(DrehWinkel,5);
        RasterZeichnen(True);
    End;
End Else

```



```

If Right Then Begin
  Right:=False;
  If DrehWinkel>-50 Then Begin
    Dec(DrehWinkel,5);
    RasterZeichnen(True);
  End;
End Else
If Up Then Begin
  Up:=False;
  If KippWinkel<60 Then Begin
    Inc(KippWinkel,5);
    RasterZeichnen(True);
  End;
End Else
If Down Then Begin
  Down:=False;
  If KippWinkel>-60 Then Begin
    Dec(KippWinkel,5);
    RasterZeichnen(True);
  End;
End;
If EndOfProg Then Begin
  M.Show(False);
  QuestionWindow('BEENDEN','Sind Sie sicher?',quest);
  If quest=False Then EndOfProg:=False;
  M.Show(True);
End;
Until EndOfProg=True;
End;
{-----Hauptprogramm-----}
Begin
  M.Reset;
  Grafik_Ein;
  FastTextInit;
  Farben_Def;
  Var_Init;
  NeighborInit;
  Screen_Init;
  M.Init;
  FeldReset;
  RasterZeichnen(True);
  M.Show(False);
  FullScreen;
  WriteCellStatus(False);
  M.Show(True);
  MenuAbfrage;
  M.Show(False);
  CloseGraph;
End.

```

ANHANG B

QUELLTEXT UNIT „MAUSMENU“

UNIT MAUSMENU;

Interface

Type GCursor=Record

```
screenMaske,cursorMaske: Array[0..15] Of Word;  
xhot,yhot: Integer;  
End;
```

Const Hand:GCursor=

```
(screenMaske:($E1FF,$E1FF,$E1FF,$E1FF,  
$E1FF,$E000,$E000,$E000,  
$0000,$0000,$0000,$0000,  
$0000,$0000,$0000,$0000);  
cursorMaske:($1E00,$1200,$1200,$1200,  
$1200,$13FF,$1249,$1249,  
$F249,$9049,$9001,$8001,  
$8001,$8001,$8001,$FFFF);  
xHot:5; yHot:0);
```

Kreuz:Gcursor=

```
(screenMaske:($FEFF,$FEFF,$FEFF,$FEFF,  
$FEFF,$FEFF,$FEFF,$0001,  
$FEFF,$FEFF,$FEFF,$FEFF,  
$FEFF,$FEFF,$FEFF,$FEFF);  
cursorMaske:($0100,$0100,$0100,$0100,  
$0100,$0100,$0100,$FFFE,  
$0100,$0100,$0100,$0100,  
$0100,$0100,$0100,$0100);  
xHot:7; yHot:7);
```

Const linkeTaste=1;

rechteTaste=2;

beideTasten=3;

keineTaste=0;

Type Maus=Object

Procedure Init;

Procedure Reset;

Function GetX:Integer;

Function GetY:Integer;

Function GetTaste:Integer;

Procedure Status(Var xpos,ypos,t:Integer);

Procedure SetXY(xpos,ypos:Integer);

Procedure Show(ch:Boolean);

Procedure MWindow(x1,y1,x2,y2:Integer);

Function Innen(x1,y1,x2,y2,xpos,ypos:Integer):Boolean;

Function Aussen(x1,y1,x2,y2,xpos,ypos:Integer):Boolean;

Procedure SetKreuz;

```

    Procedure SetHand;
End;
Menu=Object(Maus)
    Procedure Win(x1,y1,x2,y2:Integer;c1,c2,c3:Byte);
    Procedure Rand(x1,y1,x2,y2,c1,c2:Integer);
    Procedure ButtonOut(x1,y1,x2,y2:Integer;st:String);
    Procedure ButtonIn(x1,y1,x2,y2:Integer;st:String);
    Procedure GetButton(x1,y1,x2,y2,mk,xpos,ypos,taste:Integer;st:String;var chk:Boolean);
    Procedure GetSwitch(x1,y1,x2,y2,mk,xpos,ypos,taste:Integer;st1,st2:String;var chk1,chk2:Boolean);
    Procedure Window(x1,y1,x2,y2:Integer;st:Array of String; Var ov:Byte);
End;

```

Var sichtbar,KreuzChk,HandChk: Boolean;

Implementation

Uses CRT,GRAPH,DOS,FASTTEXT;

```

Var reg: Registers;
    mx,my,mt: Integer;
    p: Pointer;
    size: Word;

```

```

{----Maus initialisieren----}
Procedure Maus.Reset;
Begin
    If Mem[MemW[0:$cc+2]:MemW[0:$cc]]=$cf Then Begin
        writeln('Maus oder Maustreiber nicht gefunden!');
        Halt;
    End;
End;
{----Mausstatus (Position, Tasten) abfragen----}
Procedure Maus.status(Var xpos,ypos,t:Integer);
Begin
    reg.ax:=3;
    Intr(51,reg);
    t:=reg.bx;
    xpos:=reg.cx;
    ypos:=reg.dx;
End;
{----X-Koordinate der Mausposition einlesen----}
Function Maus.GetX:Integer;
Begin
    reg.ax:=3;
    Intr(51,reg);
    GetX:=reg.cx;
End;
{----Y-Koordinate der Mausposition einlesen----}
Function Maus.GetY:Integer;
Begin
    reg.ax:=3;
    Intr(51,reg);
    GetY:=reg.dx;

```

```

End;
{-----Maustaste einlesen-----}
Function Maus.GetTaste:Integer;
Begin
  reg.ax:=3;
  Intr(51,reg);
  GetTaste:=reg.bx;
End;
{-----Mauszeiger an bestimmte Stelle setzen-----}
Procedure Maus.SetXY(xpos,ypos:Integer);
Begin
  reg.ax:=4;
  reg.cx:=xpos;
  reg.dx:=ypos;
  Intr(51,reg);
End;
{-----Mauszeiger ein/ausschalten-----}
Procedure Maus.Show(ch:Boolean);
Begin
  If ch And Not sichtbar Then Begin
    reg.ax:=1;
    sichtbar:=True;
    Intr(51,reg);
  End;
  If Not ch And sichtbar Then Begin
    reg.ax:=2;
    sichtbar:=False;
    Intr(51,reg);
  End;
End;
{-----Funktion zur Ermittlung der kleineren Variable-----}
Function min(a,b:Integer):Integer;
Begin
  If a<b Then min:=a
  Else min:=b;
End;
{-----Funktion zur Ermittlung der größeren Variable-----}
Function max(a,b:Integer):Integer;
Begin
  If a>b Then max:=a
  Else max:=b;
End;
{-----Werte der Variablen tauschen-----}
Procedure SwapXY(Var a,b:Integer);
Var h: Integer;
Begin
  h:=a;
  a:=b;
  b:=h;
End;
{-----Mausfenster setzen-----}
Procedure Maus.MWindow(x1,y1,x2,y2:Integer);
Begin

```

```

reg.ax:=7;
reg.cx:=min(x1,x2);
reg.dx:=max(x1,x2);
Intr(51,reg);
reg.ax:=8;
reg.cx:=min(y1,y2);
reg.dx:=max(y1,y2);
Intr(51,reg);
End;
{----Abfrage, ob Maus sich innerhalb eines best. Bereichs befindet----}
Function Maus.Innen(x1,y1,x2,y2:Integer;xpos,ypos:Integer):Boolean;
Begin
  If x1>x2 Then SwapXY(x1,x2);
  If y1>y2 Then SwapXY(y1,y2);
  If (xpos>=x1) And (xpos<=x2) And (ypos>=y1) And (ypos<=y2) Then
    Innen:=True
  Else
    Innen:=False;
End;
{----Abfrage, ob Maus sich außerhalb eines best. Bereichs befindet----}
Function Maus.Aussen(x1,y1,x2,y2:Integer;xpos,ypos:Integer):Boolean;
Begin
  If x1>x2 Then SwapXY(x1,x2);
  If y1>y2 Then SwapXY(y1,y2);
  If (xpos<x1) Or (xpos>x2) Or (ypos<y1) Or (ypos>y2) Then
    aussen:=True
  Else
    aussen:=False;
End;
{----Hand-Mauszeiger setzen----}
Procedure Maus.SetHand;
Begin
  If KreuzChk And (HandChk=False) Then Begin
    reg.ax:=9;
    reg.bx:=hand.xhot;
    reg.cx:=hand.yhot;
    reg.dx:=ofs(hand.screenMaske);
    reg.es:=seg(hand.screenMaske);
    Intr(51,reg);
    KreuzChk:=False;
    HandChk:=True;
  End;
End;
{----Kreuz-Mauszeiger setzen----}
Procedure Maus.SetKreuz;
Begin
  If HandChk And (KreuzChk=False) Then Begin
    reg.ax:=9;
    reg.bx:=kreuz.xhot;
    reg.cx:=kreuz.yhot;
    reg.dx:=ofs(kreuz.screenMaske);
    reg.es:=seg(kreuz.screenMaske);
    Intr(51,reg);
  End;
End;

```

```

    HandChk:=False;
    KreuzChk:=True;
End;
End;
{-----Mausfenster und Mauszeiger setzen-----}
Procedure Maus.Init;
Begin
    MWindow(0,0,getmaxX,getmaxY);
    SetHand;
    SetXY(GetMaxX div 2,GetMaxY div 2);
End;
{-----Fenster mit Kanten zeichnen-----}
Procedure Menu.Win(x1,y1,x2,y2:Integer;c1,c2,c3:Byte);
Begin
    If x1>x2 Then SwapXY(x1,x2);
    If y1>y2 Then SwapXY(y1,y2);
    SetFillStyle(1,c2);
    SolidBar(x1,y1,x2,y2);
    SetColor(c1);
    Line(x1,y1,x2,y1);
    Line(x1+1,y1+1,x2-1,y1+1);
    Line(x1,y1,x1,y2);
    Line(x1+1,y1+1,x1+1,y2-1);
    SetColor(c3);
    Line(x2,y1,x2,y2);
    Line(x2-1,y1+1,x2-1,y2-1);
    Line(x1,y2,x2,y2);
    Line(x1+1,y2-1,x2-1,y2-1);
End;
{-----Rahmen zeichnen-----}
Procedure Rahmen(x1,y1,x2,y2:Integer);
Begin
    SetColor(15);
    Line(x1,y1,x2,y1);
    Line(x1+1,y1+1,x2-1,y1+1);
    Line(x1,y1,x1,y2);
    Line(x1+1,y1+1,x1+1,y2-1);
    Line(x2-3,y1+3,x2-3,y2-3);
    Line(x2-4,y1+4,x2-4,y2-4);
    Line(x1+3,y2-3,x2-3,y2-3);
    Line(x1+4,y2-4,x2-4,y2-4);
    SetColor(13);
    Line(x2,y1,x2,y2);
    Line(x2-1,y1+1,x2-1,y2-1);
    Line(x1,y2,x2,y2);
    Line(x1+1,y2-1,x2-1,y2-1);
    Line(x1+3,y1+3,x2-3,y1+3);
    Line(x1+4,y1+4,x2-4,y1+4);
    Line(x1+3,y1+3,x1+3,y2-3);
    Line(x1+4,y1+4,x1+4,y2-4);
End;
{-----Bildschirmausschnitt speichern-----}
Procedure GetWin(x1,y1,x2,y2:Integer);

```

```

Begin
  size:=imagesize(x1,y1,x2,y2);
  getmem(p,size);
  getimage(x1,y1,x2,y2,p^);
End;
{-----Gespeicherten Bildschirmausschnitt setzen-----}
Procedure SetWin(x1,y1:Integer);
Begin
  PutImage(x1,y1,p^,0);
  FreeMem(p,size);
End;
{-----Schaltfläche darstellen-----}
Procedure Menu.ButtonOut(x1,y1,x2,y2:Integer;st:String);
Var tx,ty: Integer;
Begin
  Setfillstyle(1,14);
  SolidBar(x1,y1,x2,y2);
  SetColor(15);
  Line(x1,y1,x2,y1);
  Line(x1,y1,x1,y2);
  SetColor(13);
  Line(x2,y1,x2,y2);
  Line(x1,y2,x2,y2);
  tx:=x1+(x2-x1-TextWidth(st)) div 2;
  ty:=y1+(y2-y1-TextHeight(st)) div 2;
  SetColor(0);
  ShowTextXY(tx+1,ty+1,st);
  SetColor(15);
  ShowTextXY(tx,ty,st);
End;
{-----Gedrückte Schaltfläche darstellen-----}
Procedure Menu.ButtonIn(x1,y1,x2,y2:Integer;st:String);
var tx,ty:Integer;
Begin
  SetFillStyle(1,14);
  SolidBar(x1,y1,x2,y2);
  SetColor(13);
  Line(x1,y1,x2,y1);
  Line(x1,y1,x1,y2);
  SetColor(15);
  Line(x2,y1,x2,y2);
  Line(x1,y2,x2,y2);
  tx:=x1+(x2-x1-TextWidth(st)) div 2;
  ty:=y1+(y2-y1-TextHeight(st)) div 2;
  SetColor(15);
  ShowTextXY(tx+2,ty+2,st);
  SetColor(0);
  ShowTextXY(tx+1,ty+1,st);
End;
{-----Rahmen des Menüfensters zeichnen-----}
Procedure Menu.Rand(x1,y1,x2,y2,c1,c2:Integer);
Begin
  SetColor(c1);

```

```

Line(x1,y1,x2,y1);
Line(x1+1,y1+1,x2-1,y1+1);
Line(x1,y1,x1,y2);
Line(x1+1,y1+1,x1+1,y2-1);
SetColor(c2);
Line(x2,y1,x2,y2);
Line(x2-1,y1+1,x2-1,y2-1);
Line(x1,y2,x2,y2);
Line(x1+1,y2-1,x2-1,y2-1);
End;
{----Mausabfrage, ob Schaltfläche (Knopf) gedrückt wird----}
Procedure Menu.GetSwitch(x1,y1,x2,y2,mk,xpos,ypos,taste:Integer;st1,st2:String;var chk1,chk2:Boolean);
Var x,y,t: Integer;
Begin
  If (taste=mk) And Innen(x1,y1,x2,y2,xpos,ypos) Then Begin
    Show(False);
    chk1:=Not Chk1;
    If Chk1 Then ButtonIn(x1+1,y1+1,x2-1,y2-1,st1);
    If (Chk1=False) Then ButtonOut(x1+1,y1+1,x2-1,y2-1,st2);
    chk2:=True;
    Show(True);
    Repeat
      Status(mx,my,mt);
    Until mt=keinetaste;
  End;
End;
{----Mausabfrage, ob Schaltfläche (Schalter) gedrückt wird----}
Procedure Menu.GetButton(x1,y1,x2,y2,mk,xpos,ypos,taste:Integer;st:String;var chk:Boolean);
Var x,y,t: Integer;
Begin
  If (taste=mk) And Innen(x1,y1,x2,y2,xpos,ypos) Then Begin
    Show(False);
    ButtonIn(x1,y1,x2,y2,st);
    Show(True);
    Repeat
      Status(x,y,t);
    Until (t=keinetaste) or aussen(x1,y1,x2,y2,x,y);
    Show(False);
    ButtonOut(x1,y1,x2,y2,st);
    Show(True);
    If Innen(x1,y1,x2,y2,x,y) Then chk:=True
      Else chk:=False;
  End;
End;
{----Menüfenster mit Liste von Befehlen darstellen----}
Procedure Menu.Window(x1,y1,x2,y2:Integer;st:Array of String;Var ov:Byte);
Var stranzahl,th,tb: Byte;
  i: Integer;
  b: Array[0..10] Of Boolean;
  Ende: Boolean;
Begin
  Ende:=False;
  For i:= 0 To 10 Do

```



```

b[i]:=False;
If x1>x2 Then SwapXY(x1,x2);
If y1>y2 Then SwapXY(y1,y2);
stranzahl:=sizeof(st) div sizeof(String);
Show(False);
GetWin(x1,y1,x2,y2);
SetFillStyle(1,14);
SetColor(14);
SolidBar(x1,y1,x2,y2);
Rahmen(x1,y1,x2,y2);
For i:=0 To stranzahl-1 Do
ButtonOut(x1+6,y1+6+i*20,x2-7,y1+23+i*20,st[i]);
Show(True);
Repeat
  Status(mx,my,mt);
  For i:= 0 To stranzahl-1 Do Begin
    If (mt<>0) And Aussen(x1,y1,x2,y2,mx,my) Then Ende:=True;
    Getbutton(x1+6,y1+6+i*20,x2-7,y1+23+i*20,linketaste,mx,my,mt,st[i],b[i]);
  End;
  For i:= 0 To stranzahl-1 Do
    If b[i] Then Begin
      ov:=i+1;
      Ende:=True;
    End;
  Until Ende=True;
  Show(False);
  SetWin(x1,y1);
  Show(True);
End;

Begin
sichtbar:=False;
HandChk:=False;
KreuzChk:=True;
End.

```

ANHANG C

QUELLTEXT UNIT „FASTTEXT“¹⁵

UNIT FASTTEXT;

Interface

```
Procedure ShowText(str:String);
Procedure ShowTextXY(x,y:Integer;str:String);
Procedure SolidBar(x1,y1,x2,y2:Integer);
Procedure InitFast;
Procedure GetCharacters;
Procedure SetCharMem(Var ch);
Procedure SetVga;
Procedure ResetVga;
Procedure SetHotColor(c:Integer);
Function GetHotColor:Integer;
```

Implementation

Uses GRAPH,DOS;

```
Var keyfarbe: Byte;
    chars: Pointer;
```

```
{-----Buchstaben einlesen-----}
```

```
Procedure GetCharacters;
```

```
Var i: Integer;
```

```
Begin
```

```
  Setcolor(1);
```

```
  SetFillStyle(solidfill,black);
```

```
  SetPalette(1,black);
```

```
  i:=0;
```

```
  While i<>2048 Do Begin
```

```
    MoveTo(0,0);
```

```
    OutText(chr(i div 8));
```

```
    Asm
```

```
      push ds
```

```
      lds si,chars
```

```
      mov ax,$a000
```

```
      mov es,ax
```

```
      add si,l
```

```
      mov di,0
```

```
      mov ax,es:[di]
```

```
      mov [si],ax
```

```
      add di,80
```

```
      inc si
```

```
      mov ax,es:[di]
```

¹⁵EBERLE, Klaus: FastVGA.PAS. In: Sonderausgabe Dos Extra Nr.19, S.90-98.

```

    mov [si],ax
    add di,80
    inc si
    mov ax,es:[di]
    mov [si],ax
    add di,80
    inc si
    mov ax,es:[di]
    mov [si],ax
    add di,80
    inc si
    mov ax,es:[di]
    mov [si],ax
    add di,80
    inc si
    mov ax,es:[di]
    mov [si],ax
    add di,80
    inc si
    mov ax,es:[di]
    mov [si],ax
    add di,80
    inc si
    pop ds
end;
Inc(i,8);
Bar(0,0,8,8);
End;
SetPalette(1,1);
SetFillStyle(solidfill,white);
End;
Procedure SetCharMem(Var ch);
Begin
    chars:=addr(ch);
End;
{-----Initialisierung-----}
Procedure InitFast;
Begin
    Getcharacters;
    ResetVga;
    SetHotColor(red);
End;
{-----VGA-Modus initialisieren-----}
Procedure SetVga;
Var r: Registers;
Begin
    r.ah:=0;
    r.al:=$12;
    Intr($10,r);

```

```

End;
Procedure ResetVga;
Begin
  Asm
    mov dx,$3ce
    mov ax,$ff08
    out dx,ax
    mov ax,$0005
    out dx,ax
    mov ax,$0003
    out dx,ax
    mov ax,$0000
    out dx,ax
    mov ax,$0001
    out dx,ax;
  End;
End;
{-----Farbe für HotKey vergeben-----}
Procedure Sethotcolor(c:Integer);
Begin
  keyfarbe:=c;
End;
{-----Farbe des HotKey einlesen-----}
Function Gethotcolor:Integer;
Begin
  GetHotColor:=keyfarbe;
End;
{-----Prozedur für schneller Textausgabe im Grafikmodus-----}
Procedure Showtext(str:String);
Const hoehe=8;
Var limits: ViewPortType;
    farbe: Byte;
    x,y: Integer;
Begin
  If length(str)<>0 Then Begin
    GetViewSettings(limits);
    farbe:=getcolor;
    x:=getx;
    y:=gety;
    MoveTo(x+length(str)*8,y);
    Asm
      mov dx,$3ce
      mov ax,$0005
      out dx,ax
      mov al,$03
      out dx,ax
      mov ax,$0f01
      out dx,ax
      xor al,al
      mov ah,farbe
      out dx,ax
      lea di,Str
      mov dl,ss:[di]
    End;
  End;
End;

```

```

mov ah,$a0
mov es,ax
mov ax,y
add ax,limits.y1
cmp ax,480
jae @ende
mov bx,ax
mov cl,6
shl bx,cl
mov cl,4
shl ax,cl
add bx,ax
mov ax,x
add ax,limits.x1
cmp ax,640
jae @ende
mov cl,al
shr ax,1
shr ax,1
shr ax,1
add bx,ax
inc di
@nextchar:
mov al,ss:[di]
cmp al,'{'
jne @nextzeichen
dec di
push dx
mov dx,$3ce
xor al,al
mov ah,keyfarbe
out dx,ax
pop dx
inc di
mov al,ss:[di]
@nextzeichen:
xor ah,ah
shl ax,1
shl ax,1
shl ax,1
push dx
push bx
push di
push ds
lds si,Chars
add si,ax
and cl,7
mov dx,$3ce
mov ch,hoehe
@nextbyte:
mov di,cx
mov ah,[si]
xor al,al

```

```

shr ax,cl
mov ch,al
mov al,08
out dx,ax
mov ah,es:[bx]
mov es:[bx],ah
mov ah,ch
out dx,ax
inc bx
mov ah,es:[bx]
mov es:[bx],ah
add bx,79
inc si
mov cx,di
dec ch
jnz @nextbyte
pop ds
pop di
pop bx
inc bx
inc di
segss mov al,[di]
cmp al,','
jne @nochhot
xor al,al
mov ah,farbe
out dx,ax
inc di
pop dx
dec dl
push dx
@nochhot:
pop dx
dec dl
jnz @nextchar
@ende:
mov dx,$3ce
mov ax,$0001
out dx,ax
xor al,al
out dx,ax
End;
End;
End;
{----Textausgabe an bestimmter Position----}
Procedure ShowTextXY(x,y:Integer;str:String);
Begin
  MoveTo(x,y);
  ShowText(str);
End;
{----Verbesserte Prozedur für Ausgabe eines gefüllten Rechtecks am Bildschirm----}
Procedure SolidBar(x1,y1,x2,y2:Integer);
Var startmask,endmask: Byte;

```

```

    filler: FillSettingsType;
    limits: ViewPortType;
Begin
  GetViewSettings(limits);
  GetFillSettings(filler);
  Asm
    mov ax,x2
    cmp ax,x1
    jae @xok
    mov dx,x1
    mov x1,ax
    mov x2,dx
    @xok:
    mov ax,y2
    cmp ax,y1
    jae @yok
    mov dx,y1
    mov y1,ax
    mov y2,dx
    @yok:
    mov ax,y1
    add ax,limits.y1
    cmp ax,480
    jae @ende
    mov di,ax
    mov cl,6
    shl di,cl
    mov cl,4
    shl ax,cl
    add di,ax
    mov ax,x1
    add ax,limits.x1
    cmp ax,640
    jae @ende
    mov cl,al
    shr ax,1
    shr ax,1
    shr ax,1
    add di,ax
    and cl,7
    mov al,$ff
    shr al,cl
    mov startmask,al
    mov cx,x2
    add cx,limits.x1
    and cl,7
    xor cl,7
    mov al,$ff
    shl al,cl
    mov endmask,al
    mov dx,$3ce
    mov ax,$0005
    out dx,ax

```

```

mov ax,$0003
out dx,ax
xor al,al
mov ah,byte(filler.color)
out dx,ax
mov ax,$0f01
out dx,ax
push ds
mov ax,$a000
mov es,ax
mov ds,ax
mov cx,x2
add cx,limits.x1
shr cx,1
shr cx,1
shr cx,1
mov ax,x1
add ax,limits.x1
shr ax,1
shr ax,1
shr ax,1
sub cx,ax
dec cx
mov bx,y2
sub bx,y1
inc bx
mov al,$08
mov ah,startmask
out dx,ax
push di
push bx
mov si,$50
@first:
mov ah,[di]
mov [di],ah
add di,si
dec bx
jnz @first
pop bx
pop di
inc di
cmp cx,81
jae @breite0
push di
push bx
add di,cx
mov ah,endmask
out dx,ax
@last:
mov ah,[di]
mov [di],ah
add di,si
dec bx

```



```
jnz @last
pop bx
pop di
mov ah,$ff
out dx,ax
mov ds,cx
sub si,cx
cld
@zeile:
rep stosb
add di,si
mov cx,ds
dec bx
jnz @zeile
@breite0:
pop ds
@ende:
mov ax,$a001
out dx,ax
xor al,al
mov ah,0
out dx,ax
End;
End;

END.
```

ANHANG D

LITERATURVERZEICHNIS

- BARTEL, Andreas: Grafikprogrammierung mit Turbo Pascal 6.0. - Vieweg, 1992.
- DEWDNEY, A.K.: Draht- und Teppichwelten. In: Sammelband Computer Kurzweil 2, Sonderheft Spektrum der Wissenschaft, S. 139-142. - Heidelberg: Spektrum Akademischer Verlag, 1992.
- DEWDNEY, A.K.: Digitale Krümelmonster. In: Sonderheft 10, Spektrum der Wissenschaft, S. 86-89. - Heidelberg: Spektrum Akademischer Verlag, 1990.
- DEWDNEY, A.K.: Life in drei Dimensionen. In: Sammelband Computer Kurzweil, Sonderheft Spektrum der Wissenschaft, S. 192-197. - Heidelberg: Spektrum Akademischer Verlag, 1988.
- DEWDNEY, A.K.: Lineare Automaten. In: Sammelband Computer Kurzweil, Sonderheft Spektrum der Wissenschaft, S. 186-191. - Heidelberg: Spektrum Akademischer Verlag, 1988.
- DEWDNEY, A.K.: Simulierte Evolution. In: Sonderheft 10, Spektrum der Wissenschaft, S. 82-85. - Heidelberg: Spektrum Akademischer Verlag, 1990.
- EBERLE, Klaus: FastVGA.PAS. Unit schneller als BGI. In: Sonderausgabe DOS Extra Nr. 19, S. 90-98. - DMV-Verlag, 1992.
- FEKETE, Zoltán: Profi-Tools Turbo Pascal. Geometrie und Grafik. - Markt und Technik, 1990.
- GLAESER, Georg: 3D-Programmierung mit Basic. - Teubner, 1986.
- HARRINGTON, Steven: Computergrafik. Einführung durch Computergrafik. - McGraw-Hill Book Company GmbH, 1988.
- HAYES, Brian: Tabellenkalkulation. In: Sammelband Computer Kurzweil, Sonderheft Spektrum der Wissenschaft, S. 164-170. - Heidelberg: Spektrum Akademischer Verlag, 1988.
- HAYES, Brian: Zelluläre Automaten. In: Sammelband Computer Kurzweil, Sonderheft Spektrum der Wissenschaft, S. 178-185. - Heidelberg: Spektrum Akademischer Verlag, 1988.
- KASSERA Winfried; SCHRÖDER Horst: Grafik mit Turbo Pascal für IBM-PCs und Kompatible. - Markt und Technik, 1988.
- SCHMIDT Rudolf: Lehre der Perspektive und ihre Anwendung. - Bauverlag, 1987.
- WEISSENBÖCK, Martin (Hrsg.): Turbo Pascal. Borland, Version 7.0. - Wien: ADIM, Oktober 1993.

ANHANG E

ANDERE HILFSMITTEL

- Turbo Pascal 7.0: Borland
- Deluxe Paint Enhanced II: Electronic Arts, 1984,1990
- PV Pic-View-Convert: W.Wiedmann
- Microsoft Word 6.0: Microsoft Corporation, 1989-1993
- Microsoft Paintbrush: Microsoft Corporation, 1989-1993
- PC AT 286: 1 MB RAM, 40 MB HD, VGA-Adapter

ANHANG F

KALENDARIUM

- 16.09.1994: Beginn der Programmierung des Programms „Life 3D“, Erste Entwürfe für dreidimensionale Darstellung von Zellen am Bildschirm
- 21.09.1994: Anmeldung für die Fachbereichsarbeit
- 25.09.1994: Weiterentwicklung der graphischen Darstellung
- 28.09.1994: Erster Programmerversuch von zellulären Automaten in einem dreidimensionalen System
- 05.10.1994: Implementierung von Perspektive in die 3D-Darstellung
- 12.10.1994: Verbesserung der perspektivischen Darstellung
- 19.10.1994: Programmierung von Funktionen, die das Drehen und Kippen des 3D-Raumes ermöglichen.
- 26.10.1994: Programmierung der UNIT „Mausmenu“ für die Maussteuerung und das Menüsystem
- 02.11.1994: Gestaltung des Hauptbildschirms
- 09.11.1994: Implementierung der „Funktionsschaltflächen“
- 16.11.1994: Gestaltung des Zelleditors und Programmierung von Funktionen, die eine Editierung des 3D-Raumes per Maus ermöglichen.
- 23.11.1994: Fertigstellung des Hauptbildschirms
- 30.11.1994: Implementierung der Unit „Fasttext“ für eine schnellere Textausgabe im Grafikmodus
- 07.12.1994: Programmierung des Dialogfenster für die Ein- und Ausgabe von Dateien
- 14.12.1994: Implementierung eines Dialogfensters für die Konfigurierung der Zellnachbarschaften
- 21.12.1994: Beginn der Programmierung des Konfigurationseditor für das Entwerfen von Zellregeln
- 28.12.1994: Weiterentwicklung des Konfigurationseditor, jedoch anschließende Überprüfungen verlaufen teilweise negativ
- 04.01.1995: Umgestaltung der Oberfläche und Fertigstellung des modifizierten Konfigurationseditor
- 11.01.1995: Implementierung von Optionen, die eine individuelle Konfiguration der Darstellung der zellulären Gebilde erlauben
- 18.01.1995: Abschließende Test verlaufen positiv Entwerfen von Zellbildern und Erstellen von Zellregeln
- 25.01.1995: Beginn der Dokumentation des Programms

- 01.02.1995: Beginn der Beschreibung von zellulären Automaten
- 08.02.1995: Fertigstellung der Programmdokumentation
- 15.02.1995: Fertigstellung der Beschreibung von zellulären Automaten