

Prozessprogrammierung 1

Praktikum

Aufgabe: Mühle

Programmiersprache: Borland Pascal 7.0 mit der
Echtzeiterweiterung
RTKernel 4.51

Virenüberprüfung: Avira Antivir Personal
8.1.0.331

Bertram, Alexander (B_TInf 2616, 6. Fachsemester)

Inhaltsverzeichnis

1. Allgemeine Problemstellung.....	4
2. Benutzerhandbuch.....	5
2.1. Ablaufbedingungen.....	5
2.2. Programminstallation.....	5
2.3. Programmstart.....	5
2.3.1. Einstellungen.....	5
2.4. Bedienungsanleitung.....	6
2.4.1. Prozessrechner.....	6
2.4.1.1. Spielleiter.....	6
2.4.1.2. Benutzerstatistik.....	6
2.4.1.3. Benutzerstatus.....	7
2.4.1.4. CPU-Spieler-Statistik und CPU-Spieler-Status	7
2.4.1.5. Menü / Tastenbelegung.....	7
2.4.1.6. Systemstatus.....	7
2.4.2. Darstellungsrechner.....	8
2.4.2.1. Benutzerspielsteine.....	8
2.4.2.2. Spielfeld.....	8
2.4.2.3. CPU-Spielsteine.....	8
2.4.2.4. Spielablauf.....	8
2.4.2.5. Menü / Tastenbelegung.....	8
2.4.2.6. Systemstatus.....	9
2.5. Logdatei.....	9
2.6. Spielende.....	9
2.7. Fehlermeldungen.....	9
2.8. Wiederanlaufbedingungen.....	10
3. Programmierhandbuch.....	11
3.1. Taskaufteilung.....	11
3.1.1. Prozessrechner.....	11
3.1.1.1. Main.....	11
3.1.1.2. Logger.....	11
3.1.1.3. Systemstatus.....	11
3.1.1.4. IPX-Sender.....	11
3.1.1.5. IPX-Receiver.....	11
3.1.1.6. Key.....	11
3.1.1.7. Leader.....	11
3.1.1.8. Player.....	12
3.1.1.9. Steal.....	12
3.1.1.10. Token.....	12
3.1.2. Darstellungsrechner.....	13
3.1.2.1. Maintask.....	13
3.1.2.2. Logger.....	13
3.1.2.3. Systemstatus.....	13
3.1.2.4. IPX-Sender.....	13
3.1.2.5. IPX-Receiver.....	13
3.1.2.6. Board.....	13
3.1.2.7. Key.....	13
3.2. Taskumschaltung.....	13
3.2.1. Prozessrechner.....	13

3.2.2.Darstellungsrechner.....	14
3.3.Zeitauflösung.....	14
3.3.1.Prozessrechner.....	14
3.3.2.Darstellungsrechner.....	14
3.4.Prioritäten.....	14
3.4.1.Prozessrechner.....	14
3.4.2.Darstellungsrechner.....	15
3.5.Intertask-Kommunikation.....	15
3.5.1.Prozessrechner.....	16
3.5.2.Darstellungsrechner.....	16
3.6.Wichtige Datenstrukturen.....	17
3.6.1.Gemeinsam.....	17
3.6.1.1.Spielfeld.....	17
3.6.1.2.Prozessrechnernachricht.....	18
3.6.1.3.Darstellungsrechnernachricht.....	19
3.6.2.Prozessrechner.....	20
3.6.2.1.Spielleiternachricht.....	20
3.6.2.2.Spielernachricht.....	22
3.6.2.3.Spielsteinnachricht.....	24
3.6.3.Darstellungsrechner.....	25
3.6.3.1.Spielbrettnachricht.....	25

1. Allgemeine Problemstellung

Als Vorlage für die zu lösende Aufgabe steht das klassische Spiel „Mühle“.

Zur Umsetzung werden zwei Rechner verwendet, deren Prozesse über das Netzwerk kommunizieren. Auf dem einen Rechner laufen die eigentlichen Spielprozesse ab, der andere Rechner ist für die Darstellung des Spielbretts zuständig.

2. Benutzerhandbuch

2.1. Ablaufbedingungen

Für das Spiel „Mühle“ werden zwei MS DOS kompatible Rechner benötigt, die über das lokale Netzwerk per IPX kommunizieren können.

2.2. Programminstallation

Um das Spiel zu installieren, müssen die Dateien **MILLPROC.EXE** und **MILL.INI** aus dem Verzeichnis „**MILL\BIN**“ auf der CD auf die Festplatte des einen Rechners in ein Verzeichnis mit Schreibrechten kopiert werden. Dieser Rechner ist für die Spiellogik zuständig und wird im Folgenden Prozessrechner genannt.

Die Datei **MILLDISP.EXE** aus dem gleichen Verzeichnis kann, muss aber nicht, auf die Festplatte des anderen Rechners kopiert werden. Dieser Rechner ist für die Darstellung des Spielfeldes zuständig und wird im Folgenden Darstellungsrechner genannt. An diesem Rechner nimmt der Benutzer am Spielgeschehen teil.

2.3. Programmstart

Um das Spiel zu starten, muss auf dem Prozessrechner das Programm **MILLPROC.EXE** und auf dem Darstellungsrechner das Programm **MILLDISP.EXE** gestartet werden.

2.3.1. Einstellungen

Das Spiel bietet die Möglichkeit einige der Startparameter zu verändern. Diese Einstellungen werden in der Textdatei **MILL.INI** vorgenommen, die beim Spielstart geöffnet und interpretiert wird. Ist die Datei nicht vorhanden, werden Standardwerte angenommen.

Die Textdatei ist in einzelne Zeilen aufgeteilt, die den folgenden Aufbau haben:

```
#Kommentar  
Schlüssel=Wert
```

Schlüssel	Bedeutung	Wertebereich	Standardwert
FirstPlayer	Spieler, der anfängt	User oder CPU	User
UserColor	Spielsteinfarbe des menschlichen Spielers	0 - 7 (0: Schwarz, 1: Blau, 2: Grün, 3: Cyan, 4: Rot, 5: Magenta, 6: Braun, 7: Grau)	9
CPUColor	Spielsteinfarbe des CPU-Spielers		2
CPUMaxTimeLimit	Maximale Bedenkzeit des CPU-Spielers in Sekunden	0 - 10	5

2.4. Bedienungsanleitung

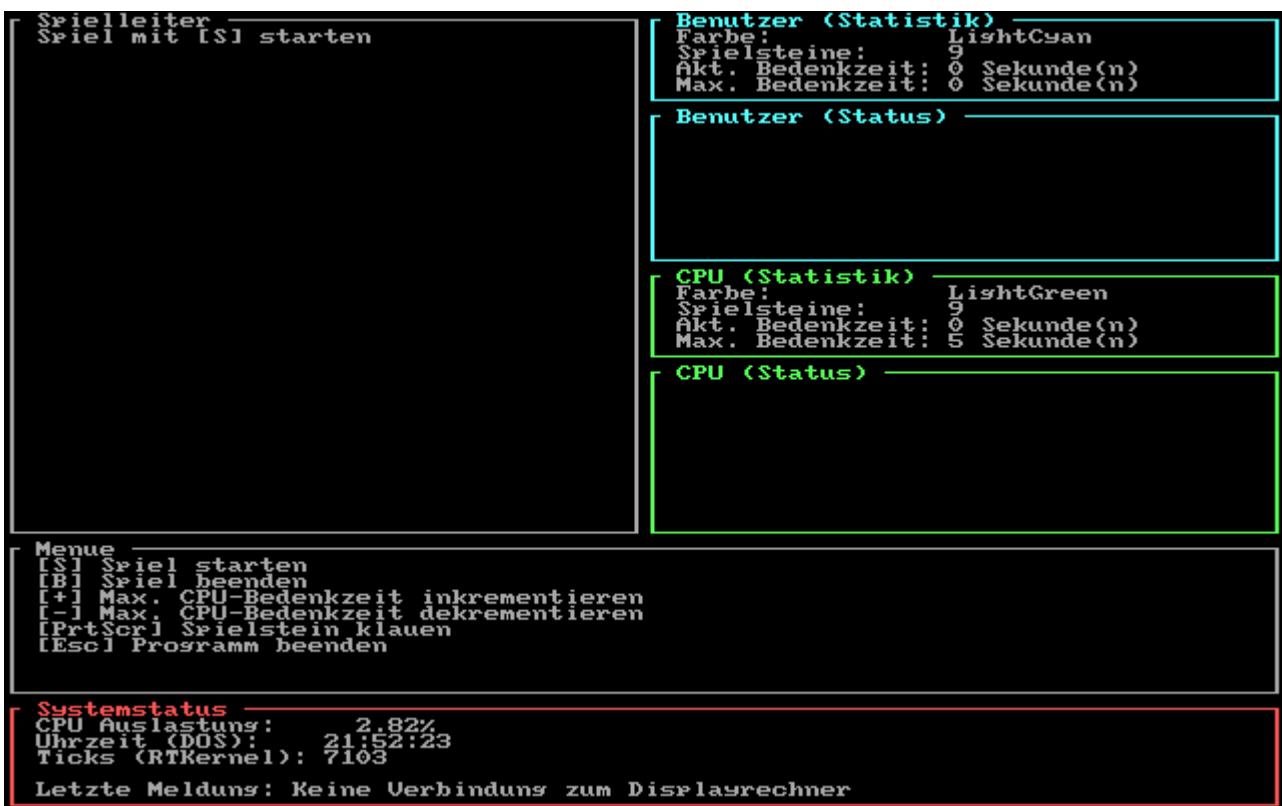
In dieser Bedienungsanleitung wird davon ausgegangen, dass der Benutzer das Spiel Mühle kennt. Der Benutzer spielt gegen einen Computergegner, die CPU. Ein virtueller Spielleiter übernimmt dabei die Leitung.

Der CPU-Spieler hat im Gegensatz zum Benutzer eine begrenzte Zeit, um seinen Zug zu wählen. Diese Zeit wird vor jedem Zug nach dem Zufallsprinzip neu bestimmt, überschreitet aber nicht einen vorgegebenen Wertebereich.

2.4.1. Prozessrechner

Der Prozessrechner ist für die Logik des Spieles zuständig.

Nach dem Programmstart ist der folgende Bildschirm zu sehen:



```
Spielleiter
Spiel mit [S] starten

Benutzer (Statistik)
Farbe: LightCyan
Spielsteine: 9
Akt. Bedenkzeit: 0 Sekunde(n)
Max. Bedenkzeit: 0 Sekunde(n)

Benutzer (Status)

CPU (Statistik)
Farbe: LightGreen
Spielsteine: 9
Akt. Bedenkzeit: 0 Sekunde(n)
Max. Bedenkzeit: 5 Sekunde(n)

CPU (Status)

Menue
[S] Spiel starten
[B] Spiel beenden
[+] Max. CPU-Bedenkzeit inkrementieren
[-] Max. CPU-Bedenkzeit dekrementieren
[PrtScr] Spielstein klauen
[Esc] Programm beenden

Systemstatus
CPU Auslastung: 2.82%
Uhrzeit (DOS): 21:52:23
Ticks (RTKernel): 7103
Letzte Meldung: Keine Verbindung zum Displayrechner
```

Abbildung 1: Startbildschirm des Prozessrechners

Der Bildschirm ist in sieben Bereiche / Fenster unterteilt: Spielleiter, Benutzerstatistik, Benutzerstatus, CPU-Spieler-Statistik, CPU-Spielers-Status, Menü und Systemstatus.

2.4.1.1. Spielleiter

In diesem Fenster werden die Aktionen des Spielleiters angezeigt.

2.4.1.2. Benutzerstatistik

Hier sind die aktuellen Infos über den menschlichen Spieler zu sehen: Farbe, Anzahl der vorhandenen Spielsteine, aktuelle Bedenkzeit und die maximale Bedenkzeit. Die aktuelle und die maximale Bedenkzeit sind beim menschlichen Spieler nicht vorhanden und sind deswegen immer

gleich Null.

2.4.1.3. Benutzerstatus

In diesem Bereich wird angezeigt, was der menschliche Spieler gerade macht.

2.4.1.4. CPU-Spieler-Statistik und CPU-Spieler-Status

Diese Fenster sind analog zu den Benutzerfenstern aufgebaut.

2.4.1.5. Menü / Tastenbelegung

Hier wird die Tastenbelegung angezeigt.

Taste	Funktion
S	Spiel starten
B	Aktuelles Spiel beenden
+	Bedenkzeit des CPU-Spielers erhöhen
-	Bedenkzeit des CPU-Spielers vermindern
PrtScr / Druck	Dem aktuellen Spieler einen Spielstein vom Spielfeld klauen
Escape	Programm beenden (mit Sicherheitsabfrage)
J	Sicherheitsabfrage mit ja beantworten
N	Sicherheitsabfrage mit nein beantworten

2.4.1.6. Systemstatus

In diesem Fenster wird der allgemeine Systemstatus angezeigt.

Neben der CPU-Auslastung und der aktuellen Uhrzeit wird hier die letzte Systemmeldung angezeigt, z. B. wenn keine Verbindung zum Darstellungsrechner aufgebaut werden konnte.

2.4.2. Darstellungsrechner

Der Darstellungsrechner visualisiert das Spielfeld. Der Benutzer nimmt hier am Spielgeschehen teil. Nach dem Programmstart ist folgender Bildschirm zu sehen:

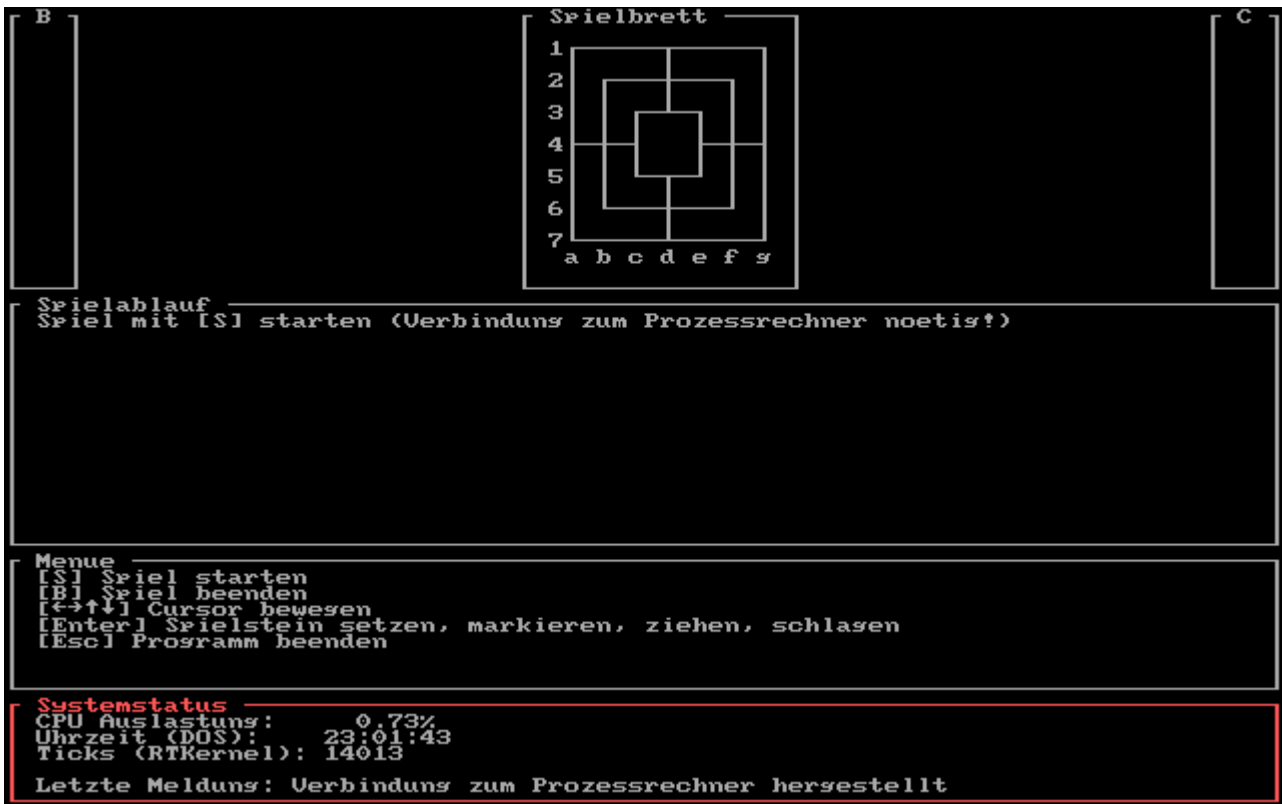


Abbildung 2: Startbildschirm des Darstellungsrechners

Der Bildschirm ist in sechs Bereiche / Fenster aufgeteilt: Benutzerspielsteine, Spielfeld, CPU-Spielsteine, Spielablauf, Menü und Systemstatus.

2.4.2.1. Benutzerspielsteine

Hier sind die nicht gesetzten Spielsteine des Benutzers zu sehen.

2.4.2.2. Spielfeld

In diesem Fenster ist das eigentliche Spielfeld dargestellt.

2.4.2.3. CPU-Spielsteine

Dieser Bereich stellt die nicht gesetzten Spielsteine des CPU-Spielers dar.

2.4.2.4. Spielablauf

Der Spielablauf ist hier in Textform dargestellt. Falsche Benutzereingaben werden farblich hervorgehoben.

2.4.2.5. Menü / Tastenbelegung

Analog zum Prozessrechner wird in diesem Fenster die Tastenbelegung angezeigt.

Taste	Funktion
S	Spiel starten
B	Aktuelles Spiel beenden
Pfeiltasten	Cursor bewegen
Enter	Spielstein setzen, markieren, ziehen, schlagen
Escape	Programm beenden (mit Sicherheitsabfrage)
J	Sicherheitsabfrage mit ja beantworten
N	Sicherheitsabfrage mit nein beantworten

2.4.2.6. Systemstatus

Dieses Fenster hat die gleiche Funktion wie das Systemstatusfenster auf dem Prozessrechner.

2.5. Logdatei

Bei jedem Programmstart wird auf dem Prozessrechner in dem Programmverzeichnis eine Logdatei erstellt, in der die Aktionen der beiden Spieler geloggt werden.

Der Aufbau eines geloggten Zuges:

Spieler	Aktion	Quellposition	Zielposition
---------	--------	---------------	--------------

Spalte	Wertebereich	Bedeutung
Spieler	['B', 'C']	Benutzer oder CPU
Aktion	['.', '*', ' ', 'x', '-']	'.': Spieler konnte nicht ziehen oder schlagen '*': Spieler hat einen Spielstein aufs Spielfeld gesetzt ' ': Spieler hat einen Spielstein von einer Quell- auf eine Zielposition gezogen 'x': Spieler hat geschlagen '-': Spieler hat eine Figur durch Schummeln verloren
Quellposition	[a(1-7) - g(1-7)]	Position, von der der Spielstein entfernt wurde
Zielposition		Position, auf die der Spielstein gesetzt wurde

2.6. Spielende

Das Spiel ist zu Ende, wenn einer der Spieler weniger als drei Figuren hat oder nicht mehr ziehen kann.

In diesem Falle muss das aktuelle Spiel mit [B] beendet werden. Dann kann ein neues Spiel mit [S] gestartet werden.

2.7. Fehlermeldungen

Meldung	Ursache	Lösung
Kein IPX-Treiber geladen.	Kein IPX-Treiber gefunden.	IPX-Treiber installieren und / oder laden.

2.8. *Wiederanlaufbedingungen*

Sollte das Spiel durch äußere Einflüsse (z. B. Stromausfall) beendet werden, so müssen die Schritte wie in 2.3 beschrieben, wiederholt werden.

3. Programmierhandbuch

3.1. Taskaufteilung

3.1.1. Prozessrechner

Auf dem Prozessrechner laufen alle Spielprozesse ab.

3.1.1.1. Main

In der Maintask werden alle anderen Tasks bis auf die Tokentasks erstellt und die Taskhandles für das Message-Passing verteilt. Zusätzlich wird hier der IPX-Knoten initialisiert, die Konfigurationsdatei eingelesen und die Konfigurationsparameter an die anderen Tasks gesendet.

Die Interruptservice-Routine für die PrtScr-Taste wird hier installiert und deinstalliert.

3.1.1.2. Logger

Die Loggertask entnimmt Nachrichten aus einer Mailbox und schreibt diese in eine Logdatei. Der Namenspräfix der Logdatei kann dabei per Message-Passing festgelegt werden. Der Namenssuffix ist eine fortlaufende Nummer. Pro Programmstart kann somit eine Logdatei erstellt werden.

Zusätzlich ist die Loggertask in der Lage Debugmeldungen und Taskinfos zu loggen.

3.1.1.3. Systemstatus

In der Systemstatustask wird im Sekundentakt die aktuelle CPU-Auslastung und die DOS-Uhrzeit ausgelesen und ausgegeben. Zusätzlich wird eine Mailbox überprüft, deren Nachrichten Fehlermeldungen wie Netzwerkfehler enthalten, die das gesamte System betreffen.

3.1.1.4. IPX-Sender

Die IPX-Sender-Task versucht eine Verbindung zum Darstellungsrechner aufzubauen. Kann die Verbindung nicht hergestellt werden, wird eine Fehlermeldung in der Systemstatus-Mailbox abgelegt. Und es wird erneut versucht die Verbindung herzustellen.

Nach erfolgreicher Verbindung wartet die Task an einer Mailbox auf Nachrichten, die per IPX an der Darstellungsrechner gesendet werden.

3.1.1.5. IPX-Receiver

Diese Task wartet an einer Mailbox auf eingehende IPX-Nachrichten, analysiert diese und leitet sie an die Zieltask weiter.

3.1.1.6. Key

Die Keytask zeigt das Menü mit der Tastenbelegung an und wartet auf einen Tastendruck. Nach einem Tastendruck wird die Taste analysiert und die entsprechende Aktion ausgeführt.

3.1.1.7. Leader

Diese Task repräsentiert den Spielleiter.

Nach dem Spielstart initialisiert die Leadertask das Spielfeld. Vor dem Ziehen wird das Spielfeld an den aktuellen Spieler und den Darstellungsrechner verschickt. Der Spieler erhält zusätzlich eine Nachricht, die ihm mitteilt, dass er seinen Zug vorbereiten kann. Als nächstes wird eine weitere Nachricht an den Spieler gesendet. Diese teilt ihm mit, dass er ziehen kann.

Hat ein Spieler gezogen, bekommt der Spielleiter eine Zugbestätigung vom Spieler. Er wertet den Zug aus, sendet diesen an den Logger und bestimmt den nächsten Spieler.

3.1.1.8. Player

Die Playertask stellt sowohl den menschlichen als auch den CPU-Spieler dar. Nach dem Erstellen der Task, wird auf eine Nachricht per Message-Passing gewartet, die die Spielerart enthält. Die Spielerart wird bei den meisten Operationen ausgewertet und es werden je nach Ergebnis unterschiedliche Aktionen ausgeführt.

Die Spielertask erstellt die Spielsteintasks, teilt den erstellten Tasks das eigene Spielertaskhandle mit und schickt den Spielsteinen eine Initialisierungsnachricht.

Vor jedem Zug sendet der Spieler an jeden seiner Spielsteine eine Anfrage nach dessen Daten (u.a. Position und Zugmöglichkeiten). Der CPU-Spieler ermittelt vor dem Zug zusätzlich seine aktuelle Bedenkzeit.

Der CPU-Spieler berechnet anhand der Zugdaten den für ihn optimalen Zug. Die Strategie dabei ist wie folgt:

1. Eigene Mühle schließen
2. Gegnerische Mühle verhindern
3. Mühle vorbereiten
4. Den ersten möglichen Zug nehmen

Der User wählt seinen Zug an dem Darstellungsrechner. Die Daten werden dann an den Prozessrechner übermittelt. Die Zugdaten werden auf dem Prozessrechner an den zu ziehenden Spielstein gesendet. Die Spielertask erwartet als Antwort vom Spielstein eine Zugbestätigung. Diese wird an den Spielleiter durchgereicht.

Das Klauen eines Steines wird auch in der Spielertask durchgeführt. Dafür wird der erste Stein gesucht und eine Nachricht an diesen Stein gesendet, dass der Stein geklaut wurde und nicht mehr im Spiel ist. Sollten keine Steine auf dem Spielfeld vorhanden sein, wird diese Nachricht ignoriert.

3.1.1.9. Steal

Die Klautask wartet an einer Semaphore auf einen Druck auf die Printscreen-Taste. Geschieht dies, so wird eine Nachricht generiert und an den Spielleiter gesendet.

3.1.1.10. Token

Die Tokentask repräsentiert einen Spielstein.

Der Spielstein berechnet nach einer entsprechenden Anfrage vom Spieler seine Zugmöglichkeiten und sendet diese wieder an den Spieler.

Empfangene Zugdaten werden an den Darstellungsrechner und die empfangene Zugbestätigung an den Spieler weitergereicht.

3.1.2. Darstellungsrechner

Der Darstellungsrechner stellt das Systemgeschehen grafisch dar.

3.1.2.1. Maintask

Die Maintask nimmt alle nötigen Initialisierungen vor, erzeugt alle Tasks und wartet auf das Ende des Programms an einer Semaphore.

3.1.2.2. Logger

Die Loggertask auf dem Darstellungsrechner wird aus dem gleichen Code wie auf dem Prozessrechner erzeugt und erfüllt somit die gleiche Funktion.

3.1.2.3. Systemstatus

Auch die Systemstatustask wird aus dem gleichen Code erzeugt wie auf dem Prozessrechner.

3.1.2.4. IPX-Sender

Die IPX-Sender-Task stellt eine Verbindung zum Prozessrechner her und wartet an einer Mailbox auf Nachrichten, die an den Prozessrechner gesendet werden sollen.

Bei einem Fehler sendet die Task eine Nachricht an die Systemstatustask, damit diese angezeigt wird.

3.1.2.5. IPX-Receiver

Empfängt der Darstellungsrechner eine Nachricht per IPX, landet diese in der Mailbox der IPX-Receiver-Task. Hier wird die Nachricht analysiert und an die entsprechende Task weitergeleitet.

3.1.2.6. Board

Die Boardtask ist für die eigentliche Darstellung des Spielfeldes, der nicht gesetzten Spielsteine und Spielablaufes zuständig.

Das Spielfeld ist mit einem Cursor ausgestattet. Dieser erlaubt die Auswahl eines Spielsteines. Ist der User an der Reihe, werden ihm die Zugmöglichkeiten des aktuell markierten Steines angezeigt. Nach der Auswahl eines Zuges wird dieser auf Gültigkeit überprüft und an den Prozessrechner gesendet. Sollte die Zugwahl ungültig sein, wird dies dem User als Fehler mitgeteilt.

3.1.2.7. Key

Die Keytask verhält sich analog zu der Keytask auf dem Prozessrechner.

Hier wird die Tastenbelegung angezeigt und Tastatureingaben entgegen genommen. Diese werden ausgewertet und an die entsprechenden Task weiter geleitet.

3.2. Taskumschaltung

3.2.1. Prozessrechner

Auf dem Prozessrechner finden preemptive Taskwechsel statt, da hier eine Interrupt Service Routine asynchron zum Spielablauf ausgelöst wird.

3.2.2. Darstellungsrechner

Auf dem Darstellungsrechner gibt es keine kritischen Zeitabschnitte, deswegen finden kooperative Taskwechsel statt.

3.3. Zeitauflösung

3.3.1. Prozessrechner

Die Zeitauflösung wurde auf 1000 Ticks pro Sekunde angepasst. Somit ist es möglich eine sekundengenaue Wartezeit zu realisieren.

3.3.2. Darstellungsrechner

Auf dem Darstellungsrechner wurde die Zeitauflösung nicht verändert, da das Programm hier im kooperativen Modus läuft.

3.4. Prioritäten

3.4.1. Prozessrechner

Task	Priorität
Maintask	MainPriority
Leadertask	MainPriority
Playertask	MainPriority + 1
Token task	MainPriority + 2
Keytask	MainPriority + 3
Stealtask	MainPriority + 3
Loggertask	MainPriority + 4
Systemstatustask	MainPriority + 4
IPXSender	MainPriority + 5
IPXReceiver	MainPriority + 5

Die IPX-Tasks haben die höchste Priorität, damit die IPX-Nachrichten sofort verarbeitet werden, sobald sie verfügbar sind.

Gleich danach kommen die Logger- und die Systemstatustasks. Diese sind hoch priorisiert, weil die Loggertask für das Ende des Programmes sorgt, indem sie ein Ereignis in der Exitsemaphore speichert. Die Systemstatustask muss sekundenweise laufen, um die Uhr und die RTKernel-Ticks zu aktualisieren.

Dahinter sind die Key- und die Stealtask. Diese verarbeiten Benutzereingaben und müssen deshalb hoch priorisiert werden.

Die Token task kommt dahinter, weil sie am Ende der Spielablaufkette auf dem Prozessrechner liegt und deshalb höher priorisiert werden muss als die Leader- und die Playertask.. Sie muss sofort nach einer Anfrage nach Spielsteindaten diese berechnen und zurück liefern.

Da ein Spieler in der Spielablaufkette zwischen Spielleiter und Spielstein liegt, ist auch die Piorität kleiner als die der Spielsteintask.

Die Priorität der Leadertask ist am kleinsten, weil der Spielleiter nur Aufgaben an die Spieler deligiert und dann auf das Ergebnis wartet.

3.4.2. Darstellungsrechner

Taskname	Priorität
Maintask	MainPriority
Boardtask	MainPriority
Keytask	MainPriority + 3
Systemstatustask	MainPriority + 4
IPXSender	MainPriority + 5
IPXReceiver	MainPriority + 5

Die Tasks auf dem Darstellungsrechner sind genauso priorisiert wie auf dem Prozessrechner.

Eine Ausnahme bildet die Boardtask. Sie hat die niedrigste Priorität, weil sie die einzige ist, die für den Spielablauf wichtig ist und nicht für die Kommunikation mit dem User oder dem Prozessrechner.

3.5. Intertask-Kommunikation

Die Intertask-Kommunikation wurde mit varianten Records realisiert, die über Mailboxen und Message-Paasing gesendet / empfangen werden.

Eine Nachricht ist wie folgt aufgebaut:

```
{ Nachrichtenart }
TMessageKind = (
    ...
);
{ Nachricht }
TMessage = record
    case Kind: TMessageKind of
        ...: (
            ...;
        );
end;
```

Die Nachrichtenart wird von der Sendertask gesetzt. Die Empfängertask kann die Nachrichtenart auswerten und anhand des Ergebnisses eine bestimmte Aktion ausführen. Diese Methode hat den Vorteil, dass jede Task nur eine Art von Nachrichten empfängt, was die Fehleranfälligkeit reduziert.

3.5.1. Prozessrechner

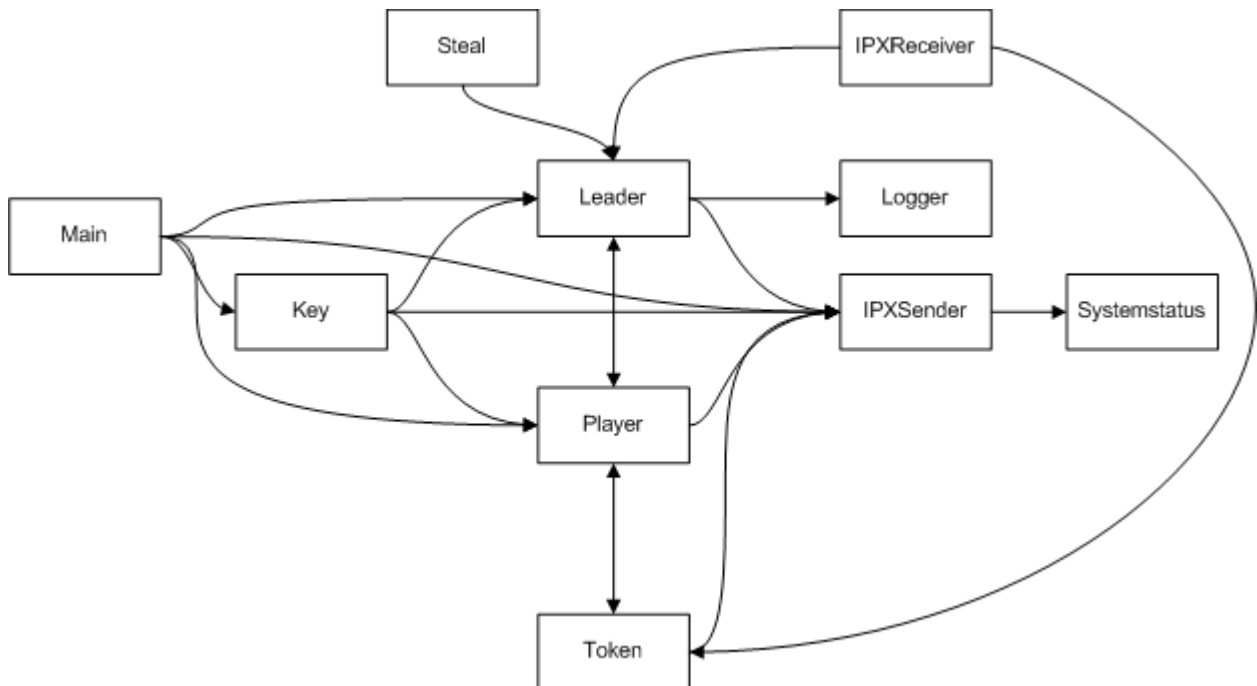


Abbildung 3: Intertask-Kommunikation auf dem Prozessrechner

Auf dem Prozessrechner gibt es eine Ausnahme der oben beschriebenen Vorgehensweise. An einer Stelle wird ein Taskhandle per Message-Passing verschickt. Und zwar braucht die Keytask das Taskhandle des CPU-Spielers, um Bedenkzeitänderungen zu ermöglichen.

3.5.2. Darstellungsrechner

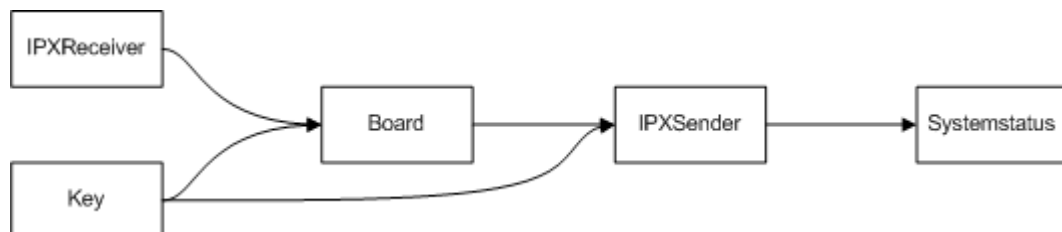


Abbildung 4: Intertask-Kommunikation auf dem Darstellungsrechner

3.6. Wichtige Datenstrukturen

3.6.1. Gemeinsam

3.6.1.1. Spielfeld

Das Spielfeld wird auf ein zweidimensionales Array abgebildet, in welchem die ungültigen Positionen markiert werden.

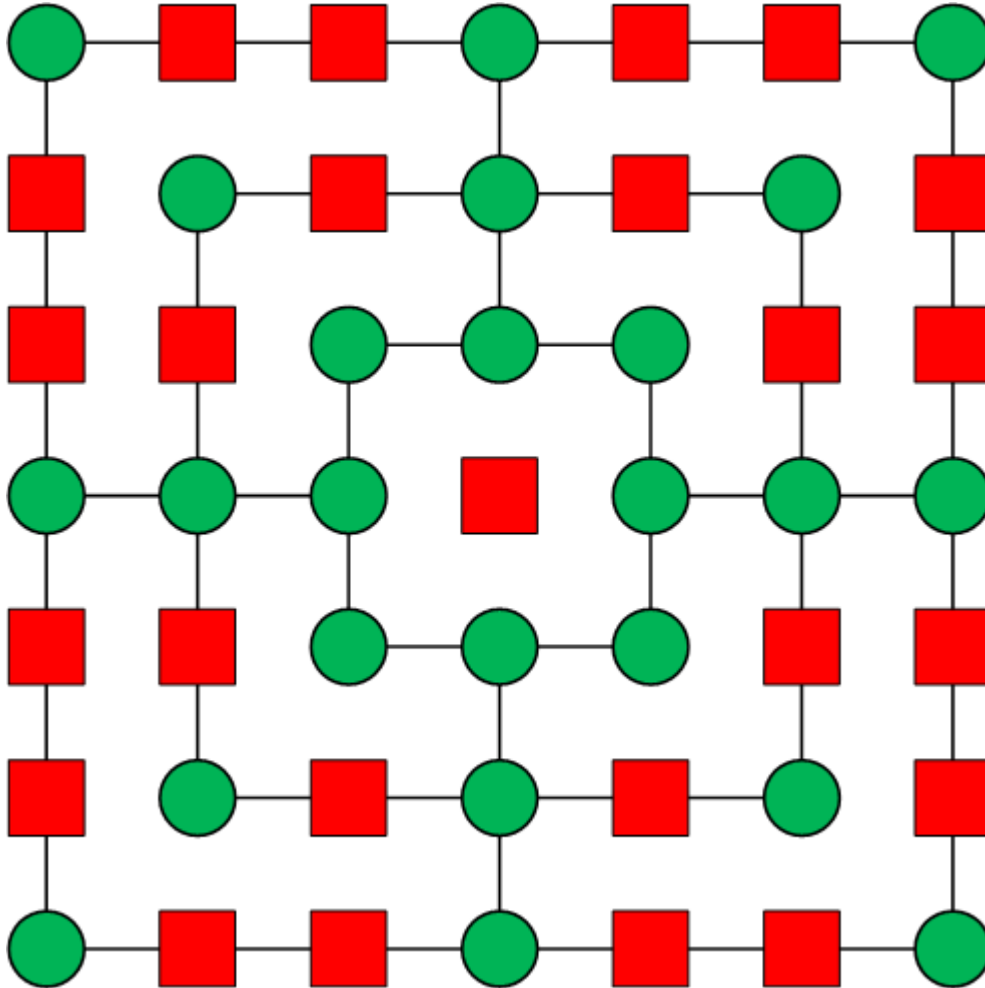


Abbildung 5: Spielfeld mit den gültig markierten (grün) Positionen

Für jede Position wird auch gespeichert, ob und wessen Spielstein drauf steht und das Taskhandle dieses Spielsteines.

```
TFieldPosProperties = record
    { Position ist gueltig }
    Valid: boolean;
    { Besitzer }
    Owner: TOwner;
    { Spielsteintaskhandle }
    TokenTH: TaskHandle;
```

end;

Zusätzlich werden in der Spielfeldstruktur die Anzahl der gesetzten und den geschlagenen Spielsteinen beider Spieler verwaltet.

```
TField = record
    { Anzahl gesetzter Spielsteine }
    PlacedTC,
    { Anzahl geschlagener Spielsteine }
    CapturedTC: TPlayerTokenCount;
    { Positionen }
    Pos: array[Low(TFieldWidth)..High(TFieldWidth) - 1,
               Low(TFieldHeight)..High(TFieldHeight) - 1] of
               TFieldPosProperties;
end;
```

Die mittlere Spalte und Zeile müssen immer gesondert verarbeitet werden, da diese logisch gesehen, aus jeweils zwei Spalten bzw. Zeilen bestehen.

3.6.1.2. Prozessrechnernachricht

Datenstruktur, die der Darstellungsrechner an den Prozessrechner sendet.

```
{ Nachrichtenart }
TProcessMessageKind = (
    { Zugwahl des Benutzers }
    prmkmUserMoveSelected,
    { Zugbestaetigung }
    prmkmTokenMoved,
    { Anfrage nach Zugmoeglichkeiten }
    prmkmGetTokenMovePossibilities,
    { Spielstart }
    prmkmStartGame,
    { Spielende }
    prmkmEndGame,
    { Programmende }
    prmkmExit);

{ Nachricht }
TProcessMessage = record
    case Kind: TProcessMessageKind of
        prmkmUserMoveSelected: (
```

```

        MoveData: TMoveData;
    );
    prmkGetTokenMovePossibilities: (
        TokenTH: TaskHandle;
    );
    prmkTokenMoved: (
        TokenMoveData: TTokenMoveData;
    );
end;

```

3.6.1.3. Darstellungsrechnernachricht

Datenstruktur, die der Prozessrechner an den Darstellungsrechner sendet.

```

{ Nachrichtart }
TDisplayMessageKind = (
    { Initialisierung }
    dmkiInit,
    { Spielerfarben }
    dmkiPlayerColors,
    { Aktueller Spieler }
    dmkiCurrentPlayer,
    { Zugmoeglichkeiten }
    dmkiMovePossibilities,
    { Spielfeld }
    dmkiField,
    { Zugdaten }
    dmkiTokenMove,
    { Spielerphase }
    dmkiPlayerStage,
    { Programmende }
    dmkiExit,
    { Spielende }
    dmkiGameOver);

{ Nachricht }
TDisplayMessage = record
    case Kind: TDisplayMessageKind of

```

```

    dmKPlayerColors: (
        Colors: TPlayerColors;
    );
    dmKCurrentPlayer: (
        Player: TPlayer;
    );
    dmKMovePossibilities: (
        Possibilities: TMovePossibilities;
    );
    dmKField: (
        Field: TField;
    );
    dmKTokenMove: (
        TokenMoveData: TTokenMoveData;
    );
    dmKPlayerStage: (
        Stage: TPlayerStage;
    );
end;

```

3.6.2. Prozessrechner

3.6.2.1. Spielleiternachricht

```

{ Nachrichtart }
TLeaderMessageKind = (
    { Spielertaskhandles }
    lemKPlayerTaskHandles,
    { Erster Spieler }
    lemKFirstPlayer,
    { Spielerfarben }
    lemKPlayerColors,
    { Initialisierung }
    lemKInit,
    { Zugvorbereitung }
    lemKPrepareMove,
    { Zug }

```

```

    lemkmoved,
    { Zugbestaetigung }
    lemkmoved,
    { Spielende }
    lemkmoved,
    { Zugwahl des Benutzers }
    lemkmoved,
    { Anfrage nach Zugmoeglichkeiten }
    lemkmoved,
    { Schlagvorbereitung }
    lemkmoved,
    { Schlag }
    lemkmoved,
    { Schlagbestaetigung }
    lemkmoved,
    { Spielende }
    lemkmoved,
    { Cheatzug }
    lemkmoved,
    { Maximale Bedenkzeit }
    lemkmoved);
{ Nachricht }
TLeaderMessage = record
    case Kind: TLeaderMessageKind of
        lemkmoved: (
            PlayerTaskHandles: TPlayerTaskHandles;
        );
        lemkmoved: (
            FirstPlayer: TPlayer;
        );
        lemkmoved: (
            PlayerColors: TPlayerColors;
        );
        lemkmoved: (
            MoveData: TMoveData;
        );

```

```

        lemKPlayerMoved, lemKPlayerCaptured: (
            TokenMoveData: TTokenMoveData;
        );
        lemKGetTokenMovePossibilities: (
            TokenTaskHandle: TaskHandle;
        );
        lemKTimeLimit: (
            TimeLimit: TTimeLimit;
        );
end;

```

3.6.2.2. Spielernachricht

```

{ Nachrichtart }
TPlayerMessageKind = (
    { Spielerart }
    plmkPlayerKind,
    { Fensterposition }
    plmkWindowPosition,
    { Fensterfarbe }
    plmkWindowColor,
    { Maximale Bedenkzeit }
    plmkMaxTimeLimit,
    { Spielleitertaskhandle }
    plmkLeaderTaskhandle,
    { Initialisierung }
    plmkInit,
    { Zug }
    plmkMove,
    { Spielsteindaten }
    plmkTokenData,
    { Inkrementierung der maximalen Bedenkzeit }
    plmkIncrementMaxTimeLimit,
    { Dekrementierung der maximalen Bedenkzeit }
    plmkDecrementMaxTimeLimit,
    { Zugwahl des Benutzers }
    plmkUserMoveSelected,

```

```

    { Zugbestaetigung des Spielsteins }
    plmkTokenMoved,
    { Anfrage nach Zugmoeglichkeiten }
    plmkGetMovePossibilities,
    { Spielfeld }
    plmkField,
    { Zugvorbereitung }
    plmkPrepareMove,
    { Schlagvorbereitung }
    plmkPrepareCapture,
    { Schlag }
    plmkCapture,
    { Cheatzug }
    plmkStealToken);
{ Nachricht }
TPlayerMessage = record
    case Kind: TPlayerMessageKind of
        plmkPlayerKind: (
            PlayerKind: TPlayer;
        );
        plmkWindowPosition: (
            WindowPosition: TWindowPosition;
        );
        plmkWindowColor: (
            WindowColor: TWindowColor;
        );
        plmkMaxTimeLimit: (
            MaxTimeLimit: TTimeLimit;
        );
        plmkLeaderTaskhandle: (
            LeaderTaskHandle: TaskHandle;
        );
        plmkTokenData: (
            TokenData: TTokenData;
        );
        plmkGetMovePossibilities: (

```

```

        TokenTaskHandle: TaskHandle;
    );
    plmkUserMoveSelected, plmkTokenMoved: (
        TokenMoveData: TTokenMoveData;
    );
    plmkField: (
        Field: TField;
    );
end;

```

3.6.2.3. Spielsteinnachricht

```

{ Spielsteinnachricht }
TTokenMessageKind = (
    { Initialisierung }
    tmkInit,
    { Spielertaskhandle }
    tmkPlayerTaskHandle,
    { Anfrage nach Spielsteindaten }
    tmkGetTokenData,
    { Zug }
    tmkMove,
    { Zugbestaetigung }
    tmkTokenMoved,
    { Spielfeld }
    tmkField,
    { Spielerphase }
    tmkPlayerStage,
    { Cheatzug }
    tmkSteal);
{ Nachricht }
TTokenMessage = record
    case Kind: TTokenMessageKind of
        tmkPlayerTaskHandle: (
            PlayerTaskHandle: TaskHandle;
        );
        tmkGetTokenData: (

```



```

        SenderTaskHandle: TaskHandle;
    );
    tmkMove: (
        MoveData: TMoveData;
    );
    tmkField: (
        Field: TField;
    );
    tmkPlayerStage: (
        PlayerStage: TPlayerStage;
    );
end;

```

3.6.3. Darstellungsrechner

3.6.3.1. *Spielbrettnachricht*

```

{ Nachrichtart }
TBoardMessageKind = (
    { Initialisierung}
    bmkInit,
    { Spielerfarben }
    bmkPlayerColors,
    { Aktueller Spieler }
    bmkCurrentPlayer,
    { Zugmoeglichkeiten }
    bmkTokenMovePossibilities,
    { Cursorbewegung }
    bmkCursorMove,
    { Spielfeld }
    bmkField,
    { Positionsmarkierung }
    bmkFieldPositionSelection,
    { Spielsteinzug }
    bmkTokenMove,
    { Spielerstatus }
    bmkPlayerStage,

```

```

        { Spielenden }
        bmkGameOver);
{ Nachricht }
TBoardMessage = record
    case Kind: TBoardMessageKind of
        bmkPlayerColors: (
            Colors: TPlayerColors;
        );
        bmkCurrentPlayer: (
            Player: TPlayer;
        );
        bmkTokenMovePossibilities: (
            TokenMovePossibilities: TMovePossibilities;
        );
        bmkCursorMove: (
            Direction: TDirection;
        );
        bmkField: (
            Field: TField;
        );
        bmkTokenMove: (
            TokenMoveData: TTokenMoveData;
        );
        bmkPlayerStage: (
            PlayerStage: TPlayerStage;
        );
    end;
end;

```