Willkommen im Reich der Würmer

IT/CT-Projekt
GFS

Snake Projekt

WormDL 1.0

---

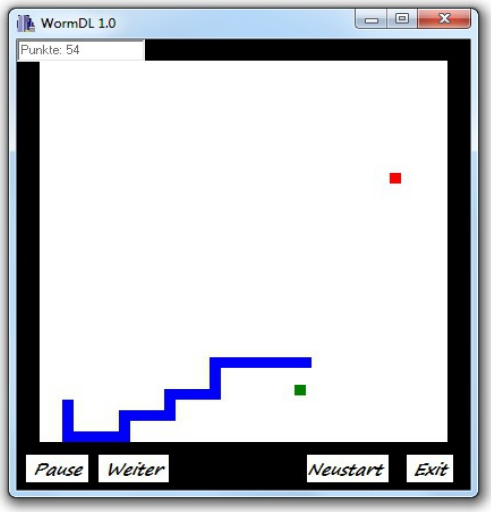
von
Dominik Kress
Lisa Schmierer

Inhaltsverzeichnis

Das Spiel

Was ist Snake allgemein?

Snake (englisch für Schlange) ist der Name eines Videospielklassikers, bei dem man eine Schlange durch ein Spielfeld steuert und Futter (manchmal Apples genannt) aufnehmen soll. Die Schlange wird mit jedem Futterhappen länger. Es können andere Schlangen auftauchen, die man nicht berühren darf. Auch können Plums auftauchen, die man nicht berühren darf (außer mit dem Schwanz). Auch Berührung der Spielfeldränder oder des eigenen Schwanzes führen zum Tod der Schlange. [wikipedia.de]



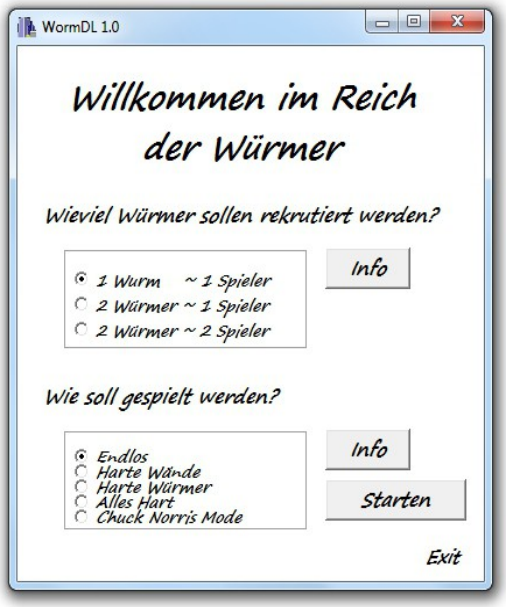
Regeln von WurmDL 1.0

Bei WurmDL 1.0 gibt es mehrere Varianten des ursprünglichen Snake. So kann man optional einstellen ob die Wände durchlässig sind, oder das Spiel bei Berührung beendet ist. Genauso ist die Regel des Spielendes, falls man den anderen Wurm, oder sich selbst anfährt optional Ein- oder Ausstellbar. Der Kontakt des anderen Wurmes, oder mit sich selbst wird zwar in jedem Falle bestraft, bei jeder Spielart (Gamemod), jedoch entweder mit dem Spielende (Gamemod Harte Wände und Harte Würmer) oder mit Punktabzug (Bei Gamemod Endlos und Harte Wände).

Das Ziel des Spieles ist es viel zu futtern und somit möglichst viele Punkte zu ergattern. Beendet ist das Spiel nur wenn je nach Spielmodus das Spiel beendet wird oder im Endlos Spiel eben erst wenn der User es selbst beendet.

Besonderheiten von WurmDL 1.0

WurmDL 1.0 lockt mit hoher Bedienerfreundlichkeit. Zu den Extras des Programmes zählt eine strukturierte Menüführung nicht nur durch ein übersichtliches Hauptmenü sondern auch ein klares Spielmenü, so dass der User stets die Kontrolle über die Einstellungen des Spieles hat.
Zudem existiert eine hohe Auswahl an verschiedenen Spielvarianten, so kann der Spieler die Wände durchgängig machen, die Kollision mit sich selbst oder optional einstellbaren anderen Würmern, die entweder von einem zweiten User, oder vom Computer als KI-Bot gesteuert werden können, lediglich gering bestrafen, oder beides entweder einzeln oder zusammen mit Spielende ahnden lassen. Zu all diesen verschiedenen Einstellungstypen des Spieles gibt es übersichtliche Informationen, die per ButtonKlick eingesehen werden können. (Siehe Bild)

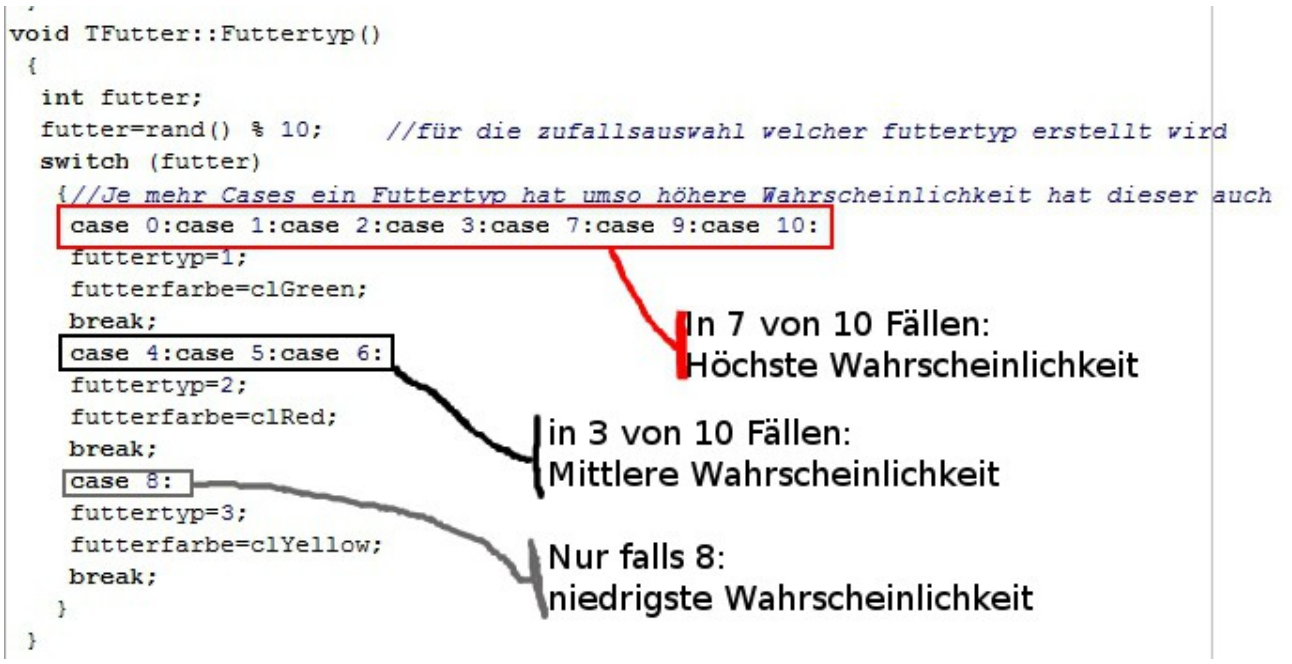


Grundlegende Eigenheiten

Am Beispiel der Futterobjekte

Wohl als eine weitere Besonderheit des Spiels kann man die Möglichkeit zählen, dass gleich bis zu 3 Futterobjekte, oder Futterpunkte auf dem Spielfeld verteilt sind. Dies wird ermöglicht durch drei Objekte der Klasse TFutter, womit wir bei Beginn des Spiels stets 3 gleiche und voneinander unabhängige Klassen erstellen auf welche wir die Futterwerte und Punkte abspeichern.
Für das Futter sind dazu die Attribute Aktiv und Typ elementar, denn diese bestimmen, ob das Futter auf dem Spielfeld gesetzt ist, oder nicht. Es bestimmt also sozusagen, wie viele Futterobjekte auf dem Spielfeld gerade fressbar sind und verhindert somit, dass ständig nur 3 Futterteile drauf warten gefressen zu werden.
Das Attribut Typ bestimmt die Art des Futters. So kann das Objekt drei verschiedene Futtertypen darstellen mit den unterschiedlichen Farben Grün, Blau und Gelb. Jeder Futtertyp hat hierbei seine eigene Wirkung sobald sie gefressen werden. So ist Grün der häufigste Typ, der die Punktzahl um 1 und die Länge ebenfalls um 1 erhöh. Rot erhöht die Punktzahl jedoch schon um 2 und die Länge um 1. Der Vorteil hierbei liegt darin, die gleiche Punktzahl erreichen zu können wie ein anderer Wurm bei kürzerer Länge und somit niedrigerer Gefahr gegen sich selbst zu kriechen.
Der dritte und unwahrscheinlichste Futtertyp mit gelber Farbe bringt dabei sogar 2 Punkte ohne Längenerweiterung.

Die Wahrscheinlichkeit hierbei wird durch eine simple kleine Funktion gestützt, so wird eine Zufallszahl von 1-10 bestimmt und in einen switch gesetzt. Je mehr Cases, also je mehr Zahlen auf den Futtertyp zutreffen, desto wahrscheinlicher ist es, dass dieser erstellt wird.
Diese unterschiedliche Wertigkeit der Futtertypen zeigt sich auch in der KI, die sich ja verständlicherweise lieber die gelben als die grünen Futterpunkte schnappen will und somit auf höhrere Futtertypen auch eine höhere Proirität legt (siehe Kapitel: KI).



Am Beispiel der KI Steuerung

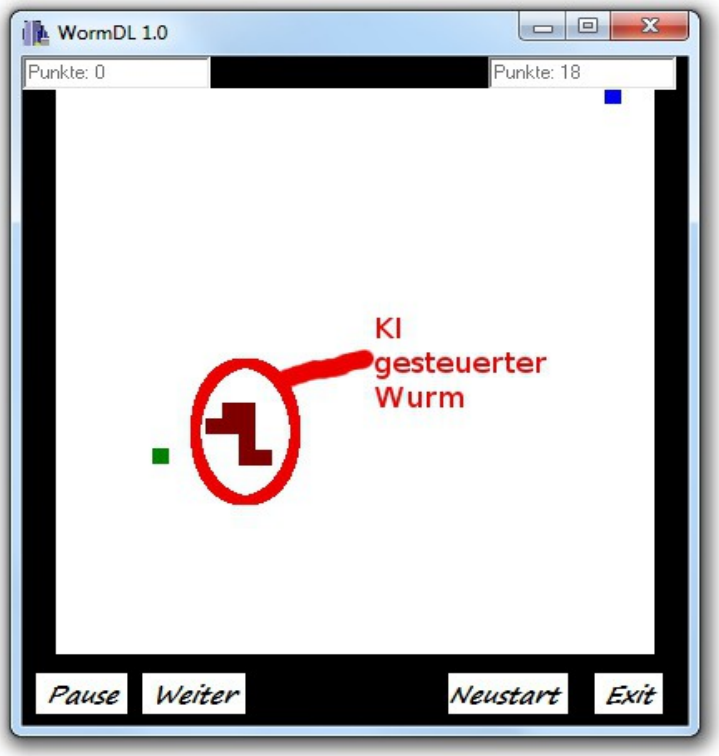
Wie der Computer steuert

Die optional zuschaltbare Künstliche Intelligenz (KI) erzeugt einen Bot-Wurm und steuert diesen automatisch. Hierbei kann die KI lediglich die Richtung des Wurmes ändern und ist somit im kompletten Vorgang des Spieles eingebunden, wie ein von einem User gesteuerter Wurm. Die Richtungszuweisung erfolgt nun lediglich nicht mehr durch Tasteneingaben, sondern durch verschiedene Abfragen, welche die KI darstellt. So wird zum Beispiel stets der sich am nächsten befindliche Futterpunkt angepeilt und die X- und Y-Werte des nahen Punktes mit denen des Wurmes verglichen um dementsprechend die Richtung zu verändern.
Dem gegenüber stellen sich einige Kontrollabfragen. So ruft ein gewisser Abzug des Abstandes je nach Futtertyp eine höhere Priorität auf diesen Futterpunkt auf, womit der Bot-Wurm also auch intelligent die Entscheidung zwischen zwei relativ gleich entfernten Punkten treffen kann.
Auch kann der Wurm bei entsprechendem Spielemodus die Abkürzungen über die Spielfeldränder intelligent nutzen: Falls also zum Beispiel ein Futterpunkt ganz am rechten Rand liegt und der Bot-Wurm am linken Rand befindlich ist, wird er durch die Wand fahren und nicht quer über das ganze Spielfeld.

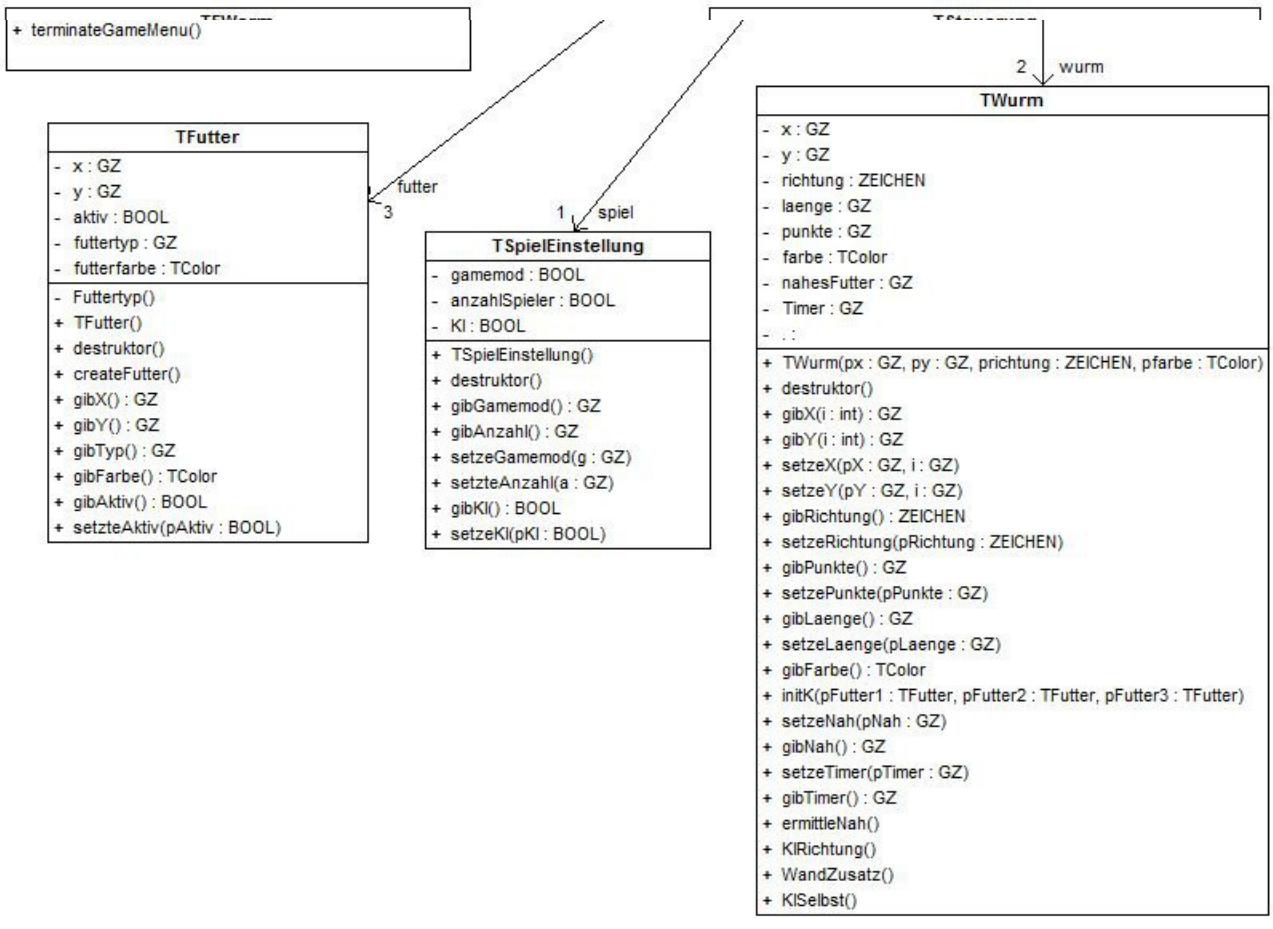
Eine letztendliche Abfrage gewährleistet dann, dass der Wurm sowohl sich selbst als auch dem anderen Wurm ausweicht, was mit einfachem Vergleich der x/y-Positionen des Kopfes und des Hindernisses funktioniert und im Falle einer bevorstehenden Kollision den Timer so ändert, dass nur noch eine andere Richtung eingeschlagen werden kann.

Zweck des Timers

Der Timer ist ein Attribut der KI, welches das größte Handicap des Wurm-Bots darstellt. Dieser zählt jedes TimerEreignis der Oberfläche hoch und erlaubt lediglich eine Auswahl der Richtung des Wurmes bei Timer<3 von oben und unten sowie bei Timer>=3 links und rechts, um einen direkteren Weg zum Futterpunkt einzuschlagen und den Weg etwas nachvollziehbarer und realistischer zu gestalten.
Dies führt zu dem unverwechselbaren Laufstil des KI Wurmes. Dieses Verhalten ist jedoch bei größerer Länge besonders Taktisch, da dann viel Länge durch häufigere Auf- und Abbewegungen gestaucht wird.

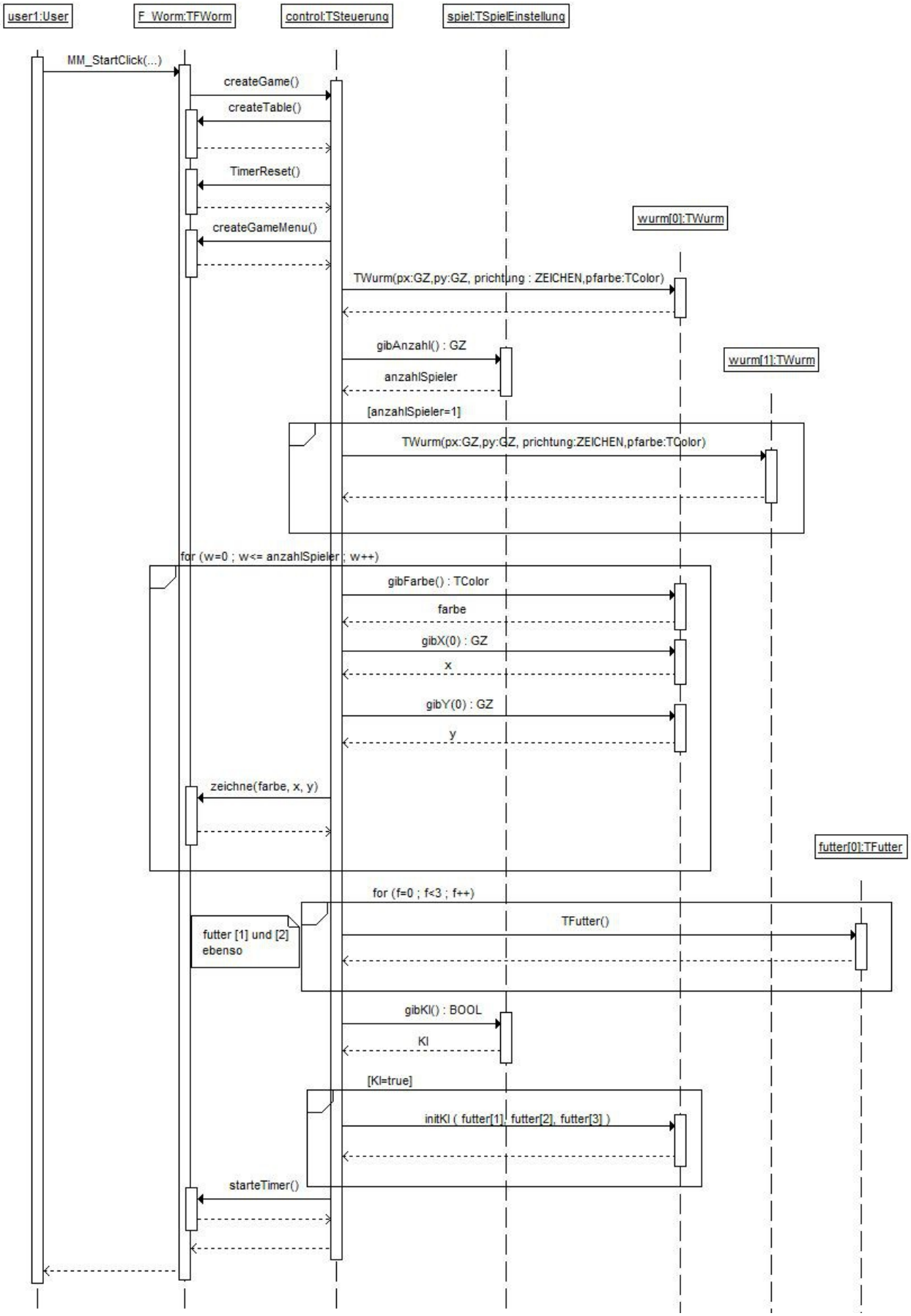

WormDL 1.0
Punkte: 0
Punkte: 18
KI
gesteuerter
Wurm
Pause
Weiter
Neustart
Exit

Exit

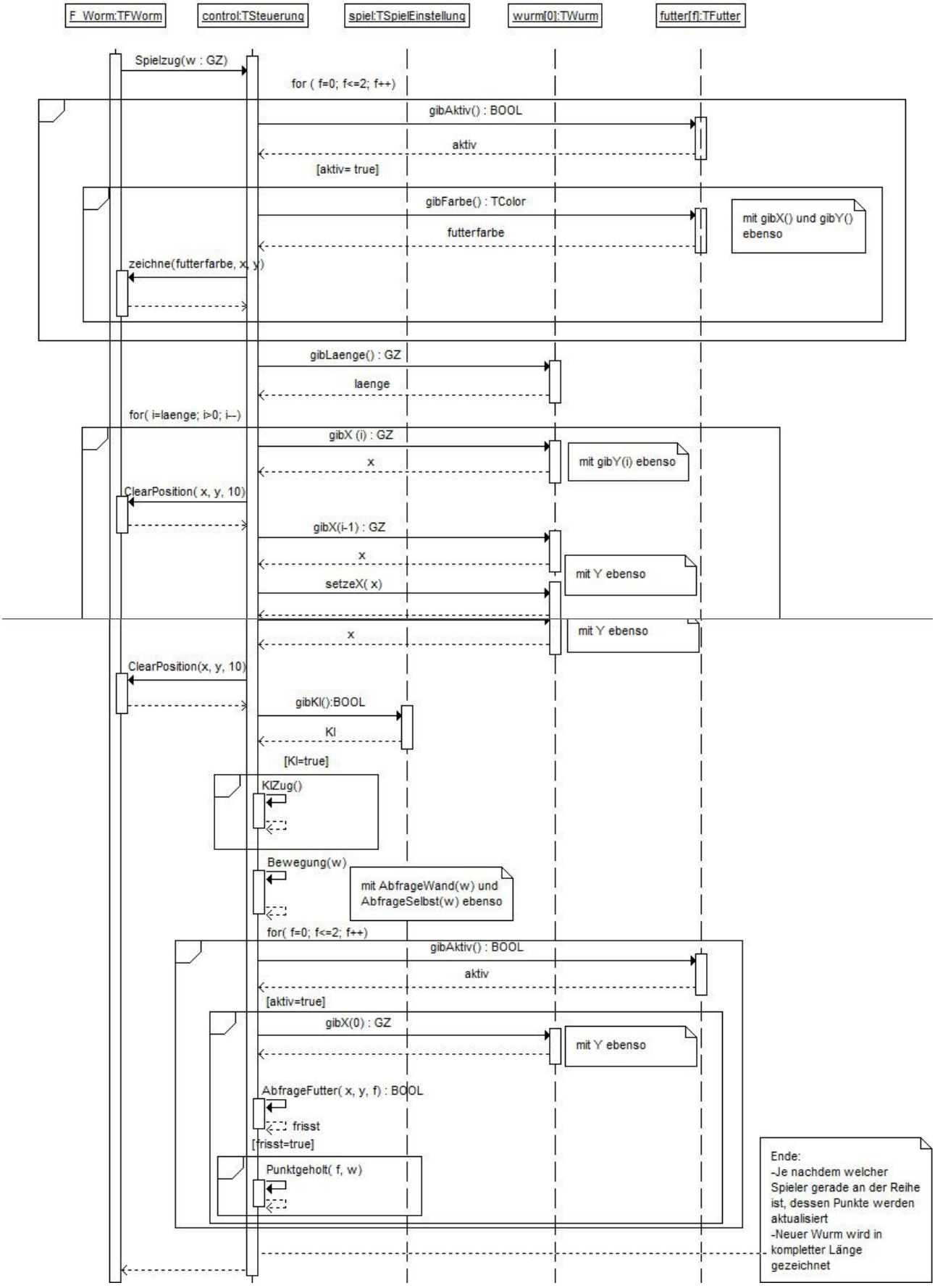

+ terminateGameMenu()
TFutter
- x : GZ
- y : GZ
- aktiv : BOOL
- futtertyp : GZ
- futterfarbe : TColor
- Futtertyp()
+ TFutter()
+ destruktor()
+ createFutter()
+ gibX() : GZ
+ gibY() : GZ
+ gibTyp() : GZ
+ gibFarbe() : TColor
+ gibAktiv() : BOOL
+ setzteAktiv(pAktiv : BOOL)
futter
3
1
spiel
TSpielEinstellung
- gamemod : BOOL
- anzahlSpieler : BOOL
- KI : BOOL
+ TSpielEinstellung()
+ destruktor()
+ gibGamemod() : GZ
+ gibAnzahl() : GZ
+ setzeGamemod(g : GZ)
+ setzteAnzahl(a : GZ)
+ gibKI() : BOOL
+ setzeKI(pKI : BOOL)
2
wurm
TWurm
- x : GZ
- y : GZ
- richtung : ZEICHEN
- laenge : GZ
- punkte : GZ
- farbe : TColor
- nahesFutter : GZ
- Timer : GZ
+ TWurm(px : GZ, py : GZ, prichtung : ZEICHEN, pfarbe : TColor)
+ destruktor()
+ gibX(i : int) : GZ
+ gibY(i : int) : GZ
+ setzeX(pX : GZ, i : GZ)
+ setzeY(pY : GZ, i : GZ)
+ gibRichtung() : ZEICHEN
+ setzeRichtung(pRichtung : ZEICHEN)
+ gibPunkte() : GZ
+ setzePunkte(pPunkte : GZ)
+ gibLaenge() : GZ
+ setzeLaenge(pLaenge : GZ)
+ gibFarbe() : TColor
+ initK(pFutter1 : TFutter, pFutter2 : TFutter, pFutter3 : TFutter)
+ setzeNah(pNah : GZ)
+ gibNah() : GZ
+ setzeTimer(pTimer : GZ)
+ gibTimer() : GZ
+ ermittleNah()
+ KIRichtung()
+ WandZusatz()
+ KISelbst()

Sequenz Diagramme

1. Für das allgemeine Szenario von MM_StartClick()



2. Für das allgemeine Szenario von MM_StartClick()



Quellenangabe und Eigenständigkeitserklärung

Eigenständigkeitserklärung

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken (dazu zählen auch Internetquellen) entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht.

_____	_____
Lisa Schmierer	Dominik Kress

Quellenangabe

wikipedia.de
www.c-plusplus.de/
de.wikibooks.org/wiki/Spezial:Suche/C%2B%2B-Programmierung/
Borland C++ Builder 5
Bouml