Geschichte des Pacemakers

Den Anstoß zur Programmierung des Pacemakers gab Wolfgang Dittmann, der zu der Zeit sich sehr mit Verteidigungsrückzügern beschäftigte. Leider habe ich die Sourcen davon nicht mehr. Sie sind durch PC-Wechsel und Diskettenvernichtung einfach nicht mehr vorhanden … Ich weiß daher auch nicht, in welchem Jahr das Programm entstand. Ich tippe da auf 2012. Es war in jedem Fall eine Zeit, in der noch mit Windows 7 gearbeitet wurde.
Windows 7 hatte den Vorteil (?), dass nur 1 Programm laufen konnte. Ich hatte daher als Programmsprachen Turbo Pascal (für die Bildschirmausgabe) und Assembler (für das eigentliche Rechenprogramm) gewählt. Beide Sprachen beherrschte ist sehr gut.

Als dann der Wunsch nach der Version Pacemaker 2 an mich herangetragen wurde, war ich bereit, das zu programmieren. Ich hatte zu der Zeit aber bislang kein neues Schachprogramm entwickelt und konnte daher nicht ahnen, was auf mich zukam…

Das Programm sollte auch unter den neuesten Versionen von Windows laufen. Da die inhaltlichen Anforderungen umfangreicher waren, kam für mich Assembler nicht mehr infrage. Ich habe mich daher für die Programmsprache C++ entschieden. Diese Sprache ist schon ziemlich alt. Sie wurde gegen Ende des 20. Jahrhunderts aus der Sprache C abgeleitet und meistens für wissenschaftliche Zwecke benutzt.
Hier ein Einschub: Wissenschaft heißt ja oft: sogenannte Langläufer, die viel Rechenzeit benötigen. Dasselbe gilt für Schachprogramme …
Es gibt verschiedene Systeme für C++. Ich habe mich für Code::Blocks entschieden. Dieses System hat eine gute Entwicklungsumgebung und bietet viele Möglichkeiten.

Die modernen Computer/PCs haben oft mehrere Prozessoren. Und prinzipiell können bei den neuen Windowsversionen mehrere Programme gleichzeitig laufen. Das erfordert natürlich, dass jedes Programm einen Bildschirm für sich allein hat. Das wird durch sogenannte Fenster realisiert. Jedes Programm hat ein solches (auch Main Window genannt). Und gleichzeitig parallel laufende andere Programme haben darauf keinen Zugriff.

In C++ als Programmiersprache gibt es aber keine Möglichkeit, ein solches Fenster zu definieren und Daten darauf auszugeben oder einzulesen ! Deshalb entstanden weitere Programme, die so etwas ermöglichen. Diese müssen dem eigenen Programm hinzugefügt werden. Hier seien mal 3 Programme/Systeme genannt:
Qt FLTK WxWidgets
Mit Qt kann man wirklich schöne Main Windows entwickeln. Aber leider musste ich feststellen, dass man mit Qt keine Langläufer entwickeln kann !! Qt kann gut genutzt werden, wenn man eine Eingabe macht, die Auswirkung im Main Window sieht und dann die nächste Eingabe macht. Bei einem Schachprogramm will man ja im Main Window nur den Fortschritt des Programms sehen.
Das Handbuch von FLTK umfasst mehr als 1200 Seiten. Nein, danke !
WxWidgets ist das für mich geeignete System. Man soll es auch in Code::Blocks integrieren können. Aber leider ist mir das noch nicht gelungen … Man findet im Internet ein paar Schulungen/Kurse die zeigen, wie man das macht. Aber die taugen nichts !!! (oder ich bin zu dumm…)

Parallel zu der Suche nach dem geeigneten System zur Schaffung eines Main-Windows habe ich schon die Programmlogik in C++ programmiert. Ich musste mir dabei das Main-Window durch Ausgaben in eine Textdatei ersetzen (die mitunter sehr groß wurde). Ich konnte dadurch aber ein paar Dinge (z.B. Widerlegungen) testen. Das Programm ist eigentlich schon ziemlich weit gekommen. Aber es fehlen noch Dinge, die einen größeren Aufwand erfordern, und ich werde diese nicht angehen, bevor nicht die Sache mit dem Main-Window und WxWidgets geklärt und funktionsfähig integriert ist.

Hier noch eine Übersicht, was das Programm schon kann und was nicht:

Schon realisiert: Typ Hoeg und Klan, Vorwärtsverteidigung, Widerlegungen auf den 1. Zug von Weiß, 3-fache Stellungswiederholung. Das Vorwärtspiel ist bisher nur als Einzüger realisiert. Für längere Vorwärtsspiele fehlen entsprechende Routinen. Sie sind aber schon eingeplant (allerdings mit niedrigster Priorität). Folgende Forderungen sind möglich: #1, h#1, s#1, dasselbe mit Patt statt Matt und ebenfalls Doppelpatt.
Folgende Märchenfiguren sind möglich: Grashüpfer, Nachtreiter, Camel, Zebra, Turmhüpfer, Läuferhüpfer, Nachtreiterhüpfer, Lion, Turmlion, Läuferlion, Leo, Pao, Vao, Equihopper verstellbar/nicht verstellbar. Weitere Figuren zu integrieren ist einfach realisierbar.

Es fehlt alles, was mit der Bildschirmausgabe (und Eingabe) zu tun hat. Wenn das Programm abgebrochen wird, dann ist derzeit keine Speicherung für ein Wiederaufsetzen möglich. Dies zu realisieren ist mit C++ eine zeitaufwendige Sache. Ebenfalls ist es nicht möglich, den kompletten Zugbaum für ein Problem auszugeben. Dafür kenne ich noch keinen Algorithmus. Und es wird garantiert sehr zeitaufwendig. Aber all das erfordert zuerst die funktionierende Verbindung zwischen C++/Code::Blocks und WxWidgets !