Simulieren mit Realbasic

von Christian Schmitz

Serie Realbasic, Folge 1 Nachdem unsere Realbasic-Serie für Einsteiger (ab Macwelt 9/2000) großen Anklang gefunden hat, wollen wir nun mit der Computer-Simulation des Spiels „Mensch ärgere Dich nicht!“ tiefer einsteigen

EINSTIEG

Realbasic ist eine leicht zu erlernende und dennoch sehr mächtige Programmiersprache für den Mac. Auch absolute Programmieranfänger erhalten nach wenigen Minuten schon ein Erfolgserlebnis. In dieser Serie führen wir den fortgeschrittenen Realbasic-Programmierer Schritt für Schritt zu einer kompletten Brettspielsimulation.

Wir beginnen mit einem neuen leeren Projekt. Vor uns sehen wir das noch leere Hauptfenster. Mit dem Menübefehl „Fenster:Eigenschaften einblenden“ blenden wir die Palette „Eigenschaften“ ein. Wir geben bei der Eigenschaft „Title“ des Fensters den Text „Mensch ärgere Dich nicht!“ ein.

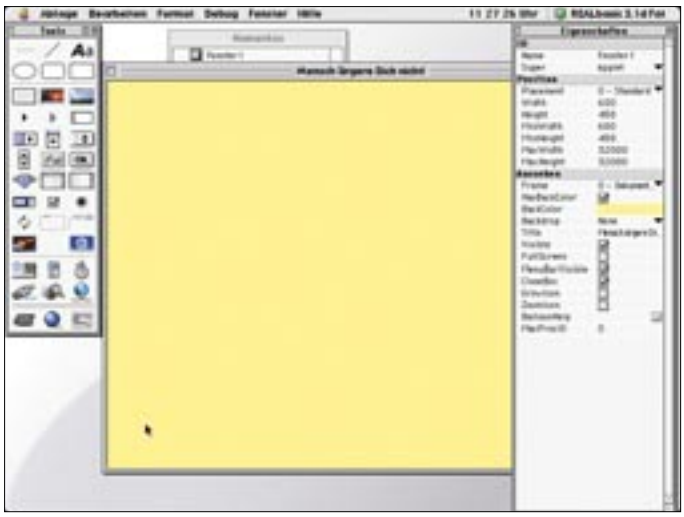
Wie bei den meisten Brettspielvarianten möchten wir den Spielfeldhintergrund gelb einfärben. Dazu klicken wir auf die Checkbox bei der Eigenschaft „HasBackColor“. Jetzt sollte der Hintergrund weiß sein. Damit er gelb wird, klicken wir auf die Farbe bei „BackColor“ und wählen im Farbdialog ein helles Gelb. Am besten im Modus „RGB“ mit 100 Prozent rot, 100 Prozent grün und 50Prozent blau.

Als Fensterbreite stellen wir 600 Pixel und als Höhe 458 Pixel ein. Dies geben wir bei den Eigenschaften Width und Height ein.

Ein Feld!

Für dieses Spiel brauchen wir verschiedene Felder auf dem Spielbrett. Zum Beispiel einen Feldtyp für den Weg, einen für das Haus und einen für das Ziel. Wir werden das Problem objektorientiert lösen. Es wird also eine Klasse für ein Feld geben, die es zeichnen soll. Alle weiteren Felder werden von dieser Klasse abgeleitet und können sich damit selber zeichnen. Wir vermeiden dadurch die Routine zum Zeichnen eines Feldes mehrmals zu programmieren. So spart man Arbeit!

Wir legen über den Menüpunkt „Ablage: Neue Klasse“ eine neue Klasse an und nennen dieses „Feld“ (Eigenschaft „Name“). Unterhalb der Eigenschaft „Name“ gibt es in der Eigenschaften-Palette die Eigenschaft „Super“, bei der wir in einem Pop-up-Menü nun den Eintrag „Canvas“ auswählen. Dadurch wird unser Feld eine Erweiterung des normalen Canvas-Steuerelements.



Spielbrett in Gelb Wie beim realen Vorbild „Mensch ärgere Dich nicht!“ geben wir dem virtuellen Spielbrett eine gelbe Farbe.

Mit einem Doppelklick auf den Namen öffnen wir die Feld-Klasse im Projektfenster und landen im Code-Editor. Unter den Events finden wir die Events der Canvas-Klasse. Diese benutzen wir fortan. Jedes Feld soll eine Farbe als Hintergrund erhalten, wobei die Felder der Häuser eine andere Farbe bekommen als die Spielfelder. Wir fügen also der Klasse eine neue Eigenschaft hinzu. Mit dem Menübefehl „Bearbeiten:Neue Eigenschaft…“ rufen wir den passenden Dialog auf und geben „Farbe as color“ ein.

Feldlein, zeichne Dich!

Es gibt in der Feld-Klasse den Event (Ereignis) „Paint“, den das System aufruft, wenn sich das Feld zeichnen soll. Das passiert zum Beispiel, wenn das Fenster zum ersten Mal erscheint, oder wenn ein anders Fenster darüber hinweg bewegt wird. Wir könnten aber auch einmal das Bedürfnis bekommen, diesem Feld sagen zu wollen, dass es sich neu zeichnen soll. Dafür wäre die Methode „Refresh“ der Control

Serie: Realbasics für Profis

1	Interface	Heft 12/2001
2	Computerspieler	Heft 01/2002
3	Mensch spielt mit	Heft 02/2002
4	Netzwerk	Heft 03/2002
5	Kompilieren und Release	Heft 04/2002

Klasse, von der das Canvas abstammt, geeignet. Wir raten aber davon ab, weil es stark flackert. Bei einem Refresh malt Realbasic zuerst den Fensterhintergrund und dann das Feld. Besser wäre, wenn wir das neue Feld direkt über das alte zeichnen. Wir benutzen daher eine eigene Methode namens „Draw“ um unser Feld zu zeichnen. Im Paint-Event schreiben wir deshalb nur

```
Draw g
```

Wir rufen dort unser Zeichnenunterprogramm auf und geben den Parameter „g“ für die Grafikumgebung weiter.
Für den Fall, dass wir selber neu zeichnen wollen, fügen wir über den Menübefehl „Bearbeiten:Neue Methode…“ eine neue Methode namens „Redraw“ ein. Diese ruft zwar auch „Draw“ auf, aber da wir kein „g“ zur Hand haben, nehmen wir die Grafikumgebung des Canvas selber. Mit der Zeile

```
Draw Graphics
```

greifen wir daher auf die Grafikumgebung unter dem Namen „Graphics“ zu.
Nun fehlt nur noch das eigentliche Unterprogramm „Draw“. Über den Menübefehl „Bearbeiten:Neue Methode…“ fügen wir ein Unterprogramm namens „Draw“ mit dem Parameter „g as graphics“ ein. Das Unterprogramm wird also in die jeweilige Grafikumgebung zeichnen. Das g im Paint-Event hat nichts mit der Graphics-Eigenschaft des Canvas zu tun! Wir übergeben dem Unterprogramm die jeweils aktuelle Umgebung.
Als erstes weisen wir mit der Anweisung

```
g.foreColor=Farbe
```

die aktuelle Zeichenfarbe auf die Farbe, die wir in der Variable „Farbe“ gespeichert haben. Mit der Zeile

```
g.fillOval 0,0,width,height
```

zeichnen wir anschließend ein gefülltes Oval in der Größe des Feldes. Die zwei Zeilen

```
g.foreColor=rgb(0,0,0)
```

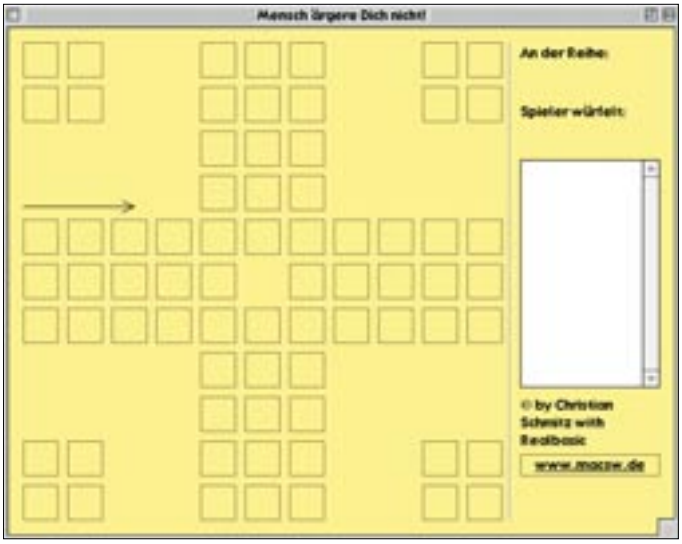
und

```
g.drawOval 0,0,width,height
```

zeichnen dann noch einen schwarzen Rand um das Feld.

Wir wollen was sehen!

Natürlich soll unser Feld auch mal in Erscheinung treten. Wir wechseln ins Hauptfenster und ziehen ein Canvas-Steuerelement aus der Palette hinein. Jetzt ändern wir die Eigenschaft mit dem Namen „Super“ von „Canvas“ auf „Feld“. Alternativ können wir auch direkt die Klasse aus dem Projektfenster in das



Spielbrett im Editor Wenn wir alle Felder korrekt positioniert haben, sollte unser Spielfeld im Editor so aussehen.

Hauptfenster ziehen. Realbasic legt dann automatisch ein neues Canvas an und ändert die Super-Klasse. Wir drücken „Befehl-R“ (Run) und sehen einen schwarzen Kreis. Schwarz, weil die Variable Farbe von dem Feld keinen Wert bekommen hat und der Defaultwert aller Farben RGB(0,0,0) also schwarz ist.

Häuslebauer

Wir schieben unser erstes Feld nach links oben bis zu den Hilfslinien. Die Left- und Top-Eigenschaften betragen dann jeweils 13. Nun ändern wir die Feldgröße auf 32 mal 32 Pixel (Width und Weight auf 32 setzen). Wir positionieren alle Felder so, dass sie bei einer Koordinate liegen, die die Ziffer drei am Ende hat. Außerdem nehmen wir einen Abstand von 40 Pixeln von einer Ecke zur nächsten. Dazu ziehen wir ein Feld mit gedrückter Alt-Taste (Option) nach rechts. Dabei kopieren wir es und schieben es so, dass es bei left = 53 liegen bleibt. Nun kopieren wir beide Felder zusammen nach unten, so dass diesmal top = 53 wird. Jetzt sollten wir vier Canvas-Steuerelemente mit der Super-Klasse „Feld“ haben. Allen geben wir den Namen „bhaus“ (für blaues Haus). Realbasic fragt uns beim Zweiten, ob wir ein Controlarray anlegen wollen. Diese Frage beantworten wir mit „ja“.

Nun haben wir vier Steuerelemente, die alle unter demselben Namen ansprechbar sind. Um sie zu unterscheiden, benutzt man einen Index. Dieser läuft hierbei von „bhaus(0)“ bis „bhaus(3)“. Realbasic zählt die Index-Eingenschaft jedes Steuerelements automatisch durch.

Wir klicken ein Feld doppelt an und landen im Code-Editor im Mousedown-Event des Feldes. Wie wir sehen, gibt es dort nur einen Eintrag für „bhaus“. Wir klicken weiter unten auf den Open-Event des Feldes. Rechts oben im Programmcode steht „sub open(index as integer)“. Realbasic ruft diesen Event vier mal auf und übergibt jedes Mal die Nummer des Feldes. Hier können wir das Feld noch vor seinem Erscheinen verändern. Das nutzen wir und fügen die Zeile

```
me.farbe=rgb(100,100,255)
```

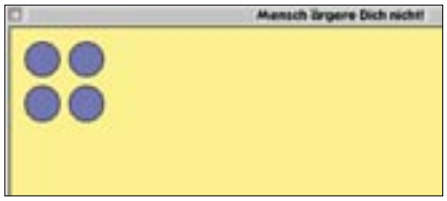
ein. Mit „me“ verweisen wir auf das jeweils aktuelle der vier Steuerelemente. „Farbe“ ist unsere Variable für die Farbe des Feldes. Wir ändern sie hier auf ein helles Blau. Wenn wir nun unser Programm starten sollten wir das unten stehende Bild sehen.

Nachbarhäuser

Nun selektieren wir die vier Felder im Hauptfenster und ziehen sie mit gedrückter Optionstaste (Alt) nach unten links in die Ecke. Damit kopieren wir alle vier Felder. Sie sollten bei der Position (left = 13, top = 373) liegen. Wir benennen vier neuen Häuser um in „rhaus“. Mit einem Doppelklick auf ein solches Feld kommen wir in den Codeeditor und ändern den Event „Open“. Da diese Felder rot werden sollen, ergänzen wir dort die Zeile

```
me.farbe=rgb(255,100,100)
```

Nun kopieren wir auf die gleiche Art alle acht Felder nach rechts, so dass die Left-Eigenschaft der am weitesten links liegenden Felder 373 beträgt. Jetzt liegen in allen



Aller Anfang ist schwer Noch ist nicht viel zu sehen auf unserem Spielfeld. Wir beginnen mit den blauen „Haus“-Feldern in der linken oberen Ecke. ►

vier Ecken jeweils vier Felder. Die Felder unten rechts nennen wir „ghaus“ für „Grünes Haus“ und die Felder oben rechts „yhaus“ für „Yellow Haus“. (yellow = englisch für Gelb). Die Farben der Felder ändern wir wie gehabt mit

```
me.farbe=rgb(100,255,100)
```

in ein Grün, beziehungsweise mit

```
me.farbe=rgb(255,255,100)
```

in ein Gelb.

Das Spielfeld

Das Spielfeld setzt sich aus 40 aneinandergelegten Feldern zusammen. Alle Felder heißen „Brett“ und sind der Reihe nach von 0 bis 39 durchnummeriert. Für Höhe und Beite nehmen wir wieder 32 Pixel. Die Felder bilden eine Art Stern mit vier Schlaufen. Anschließend fügen wir wie beim Originalspiel die Zielfelder ein. Dazu setzen wir vier Felder in jede der vier Schlaufen. Diese nennen wir analog zu den Haus-Feldern „Ziel“ (gZiel, rZiel, bZiel, yZiel).
Für die korrekten Farben schreiben wir in die Open-Events noch jeweils die Zeilen:

```
bZiel.open: me.farbe=rgb(100,100,255)
gZiel.open: me.farbe=rgb(100,255,100)
yZiel.open: me.farbe=rgb(255,255,100)
rZiel.open: me.farbe=rgb(255,100,100)
```

Dekoration im Fenster

Das Spielfeld ist noch recht kahl. Damit man weiß in welche Richtung sich die Figuren bewegen, ergänzen wir einen Richtungs-Pfeil.

Dazu ziehen wir dreimal eine Linie aus der Toolbar in unser Fenster. Passende Koordinaten für die Linien wären (13, 113, 161, 161), (102, 112, 166, 161) und (103, 113, 156, 161). Die drei Linien ergeben dann einen langen Pfeil, der nach rechts zeigt.

Den Pfeil kann man durch eine schöne Grafik ersetzen. Rechts auf die Seite des Fenster kommen noch einige Informationsanzeigen. Um sie optisch abzutrennen, nehmen wir eine Trennlinie, in Realbasic der Separator. Als Koordinaten nehmen wir left = 449, top = 13, width = 4 und height = 432. Separator-Steuerelemente sind immer vier Pixel hoch oder breit, damit das OS die Linie je nach eingestelltem Erscheinungsbild erkennen kann.

In dem abgetrennten Bereich positionieren wir oben vier statische Texte. Zunächst den Text „An der Reihe:“ und darunter ein leeres Feld mit dem Namen „spielername“. Analog dazu dann „Spieler würfelt:“ und das Feld „wurfel“.

Ergebnisse zum Mitschreiben

Jeder möchte gerne wissen, was während des Spielablaufs passiert, deshalb ergänzen wir eine Protokollfunktion, die alle Spielzüge mitschreibt. Dazu brauchen wir eine Listbox unter den Texten. Diese bekommt eine Spalte und einen vertikalen Scrollbalken (keine Überschrift). Wenn wir hierzu die rechte Listbox aus der Toolbar benutzen, sind diese Eigenschaften bereits voreingestellt, während die linke Listbox aus der Toolbar andere Einstellungen beinhaltet.

Unsere Listbox bekommt den Namen „Liste“ und die Koordinaten left = 461, top = 119, width = 126, height = 206 zugeteilt.

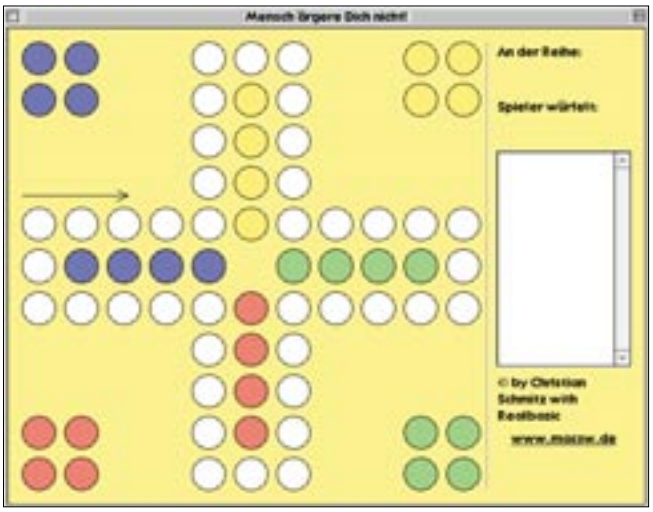
Nun brauchen wir ein Unterprogramm, das einen Text in die Protokollliste einfügt. Wir erzeugen dazu eine neue Methode namens „log“ mit dem Parameter „logtext as string“. Als einzige Codezeile ergänzen wir in dieser Methode

```
liste.insertRow 0,logtext
```

Dadurch fügt Realbasic den Text in der Liste ganz oben an der ersten Position (Realbasic zählt hier ab der Ziffer Null!) ein.
Um die Methode zu testen, können wir bei der Feld-Klasse im Event „MouseDown“ eine Zeile ergänzen

```
window1.log me.name+" "+str(me.index)
```

Wir rufen aus dem Feld heraus die Log-Methode des Fensters „window1“ auf. Als Text übergeben wir den Namen von dem Feld und daran angehängt ein Leerzeichen und die laufende Nummer des Steuerelements. Die Nummer wird als Zahl gespeichert, daher müssen wir sie erst durch die Funktion Str() in einen Text umwanden.



Spielbrett zur Laufzeit Wenn wir nach getaner Arbeit unser Programm zum Test starten, erscheint dieses fertige Spielfeld.

Aufruf | Ihr Spiel auf CD

Das in dieser Folge programmierte Interface für „Mensch ärgere Dich nicht!“ lässt sich mit eigenen Grafiken je nach Geschmack deutlich aufbessern. Wenn Sie mögen, schicken Sie uns Ihre eigene Kreation per E-Mail an: redaktion@macwelt.de Wir prämieren die gelungensten Variationen, präsentieren sie online und packen sie auf die jeweils aktuelle *Macwelt* Abo-CD!

Copyright und Link ins Web

Ein Copyright-Vermerk sichert die Rechte an dem Programm, dazu setzen wir einen statischen Text unter der Listbox. Er enthält einen Text ähnlich zu „© 2001 by Christian Schmitz und Macwelt“. Das Copyright Zeichen erreicht man mit Option-G.

Praktisch ist ein anklickbarer Link zur *Macwelt*-Homepage. Dazu benutzen wir ein Canvas mit einem darüberliegenden statischen Text. Wir ziehen das Canvas rechts unten in die Ecke und dazu einen Statiktext, so dass beide genau übereinander liegen. Der Text bekommt den Inhalt „www.macwelt.de“ und passend dazu setzen wird die Eigenschaft „underline“ auf true. Im Canvas unter dem Text schreiben wir in den „MouseDown“-Event

```
return true
```

Dadurch erklären wir dem System, dass wir uns selbst um diesen Mausklick kümmern. Genau wie bei KeyDown und MenüEvents wird der Event solange weitergereicht, bis sich jemand verantwortlich fühlt.
Im MouseUp Event ergänzen wir die Zeile

```
showurl "http://www.macwelt.de"
```

Dieser Befehl zeigt eine URL an. Das System ruft automatisch den Webbrowser auf, der im Internet-Kontrollfeld eingestellt ist.

Ausblick

In der nächsten Folge kümmern wir uns eingehend um die künstliche Intelligenz des Computergegners. *cm*