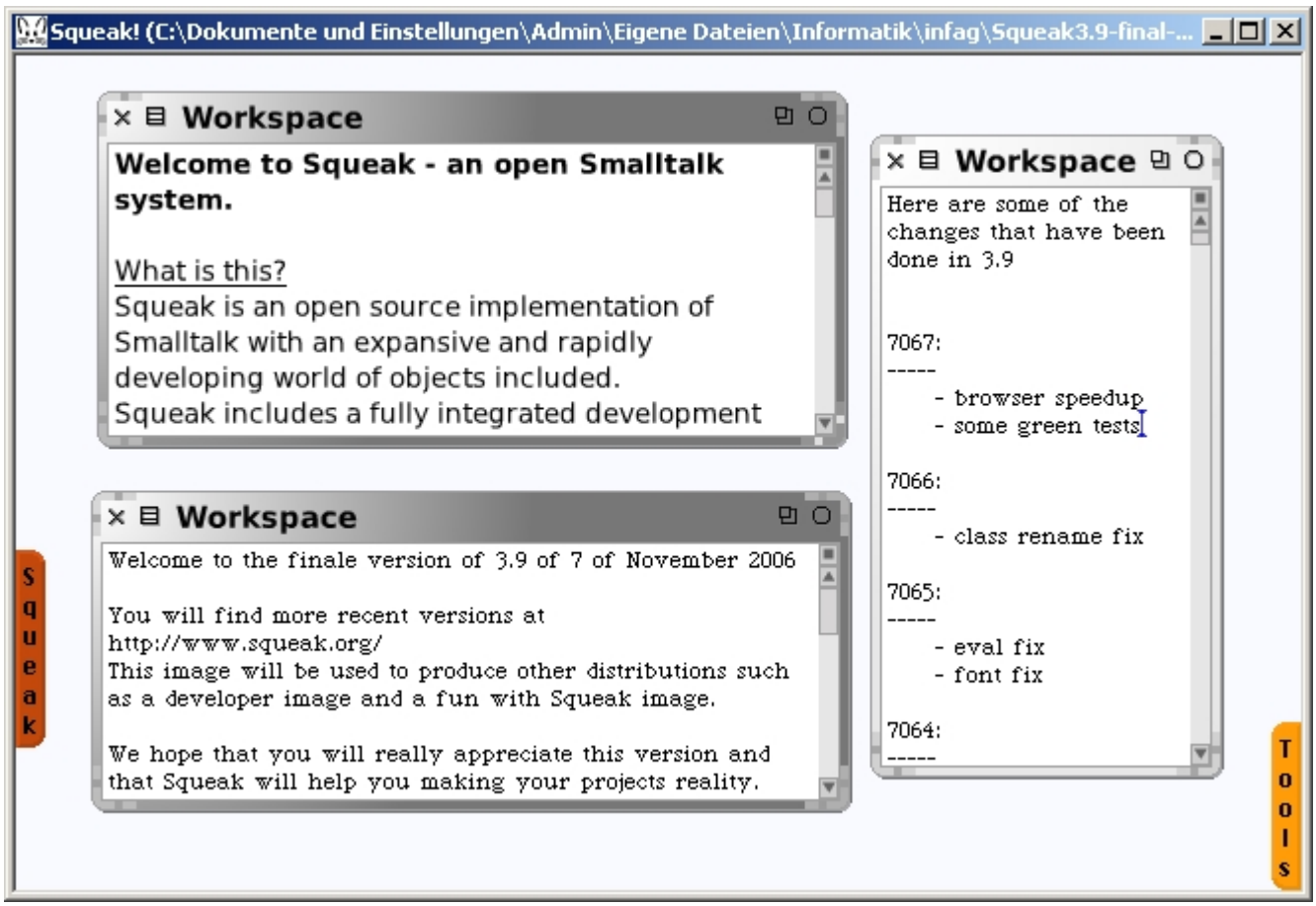
Erster Start von Squeak

In der Informatik-AG für Mädchen der Klassen 8 am Gymnasium St. Wolfhelm in Schwalmtal wird mit der Programmierumgebung Squeak gearbeitet. Es wird die Version 3.9 eingesetzt; sie kann bei www.squeak.org kostenlos bezogen werden. Der gesamte Sprachumfang liegt in einem Image. Dieses Image wird verändert, wenn nach Beenden von Squeak der Menüpunkt *Speichern und Verlassen* gewählt wird. Die Veränderungen des Image werden in der Datei mit der Endung *changes* festgehalten.

Nach dem Download und anschließendem Entpacken kann das Programm mit der Datei 

gestartet werden.

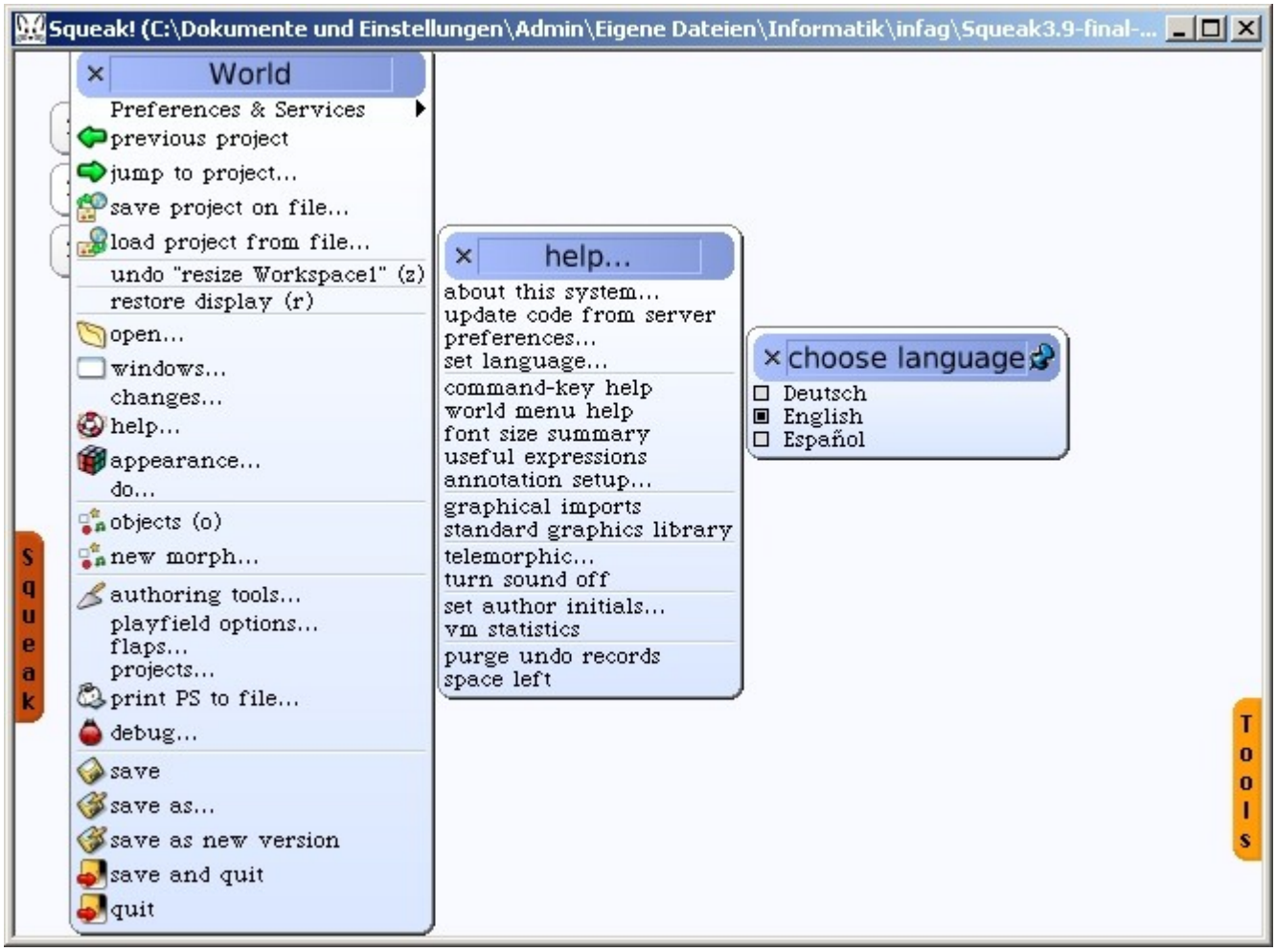


Das Startfenster zeigt drei so genannte Workspaces mit Informationen über die Programmiersprache Squeak. Demnach ist Squeak eine kostenlose OpenSource Implementation der Programmiersprache Smalltalk für alle gängigen Plattformen. Sie wurde von Alan Key unter der Mitarbeit von Seymour Papert, Jerome Brunner und vielen anderen entwickelt.

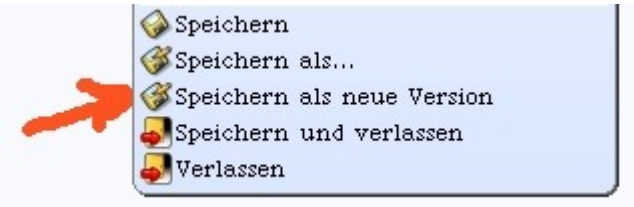
Squeak beinhaltet eine vollständige Programmierumgebung für Smalltalk. Nach kurzer Zeit ist auch ein Anfänger in der Lage ansprechende Programme konstruktiv zu entwickeln.

Nach dem ersten Start von Squeak wird die Sprache Deutsch eingestellt. Man ruft mit einem Linksklick außerhalb eines Workspaces das Weltmenü auf und wählt unter Hilfe (*help...*) den Menüpunkt Sprache (*set language...*) auf.

Taschenrechner

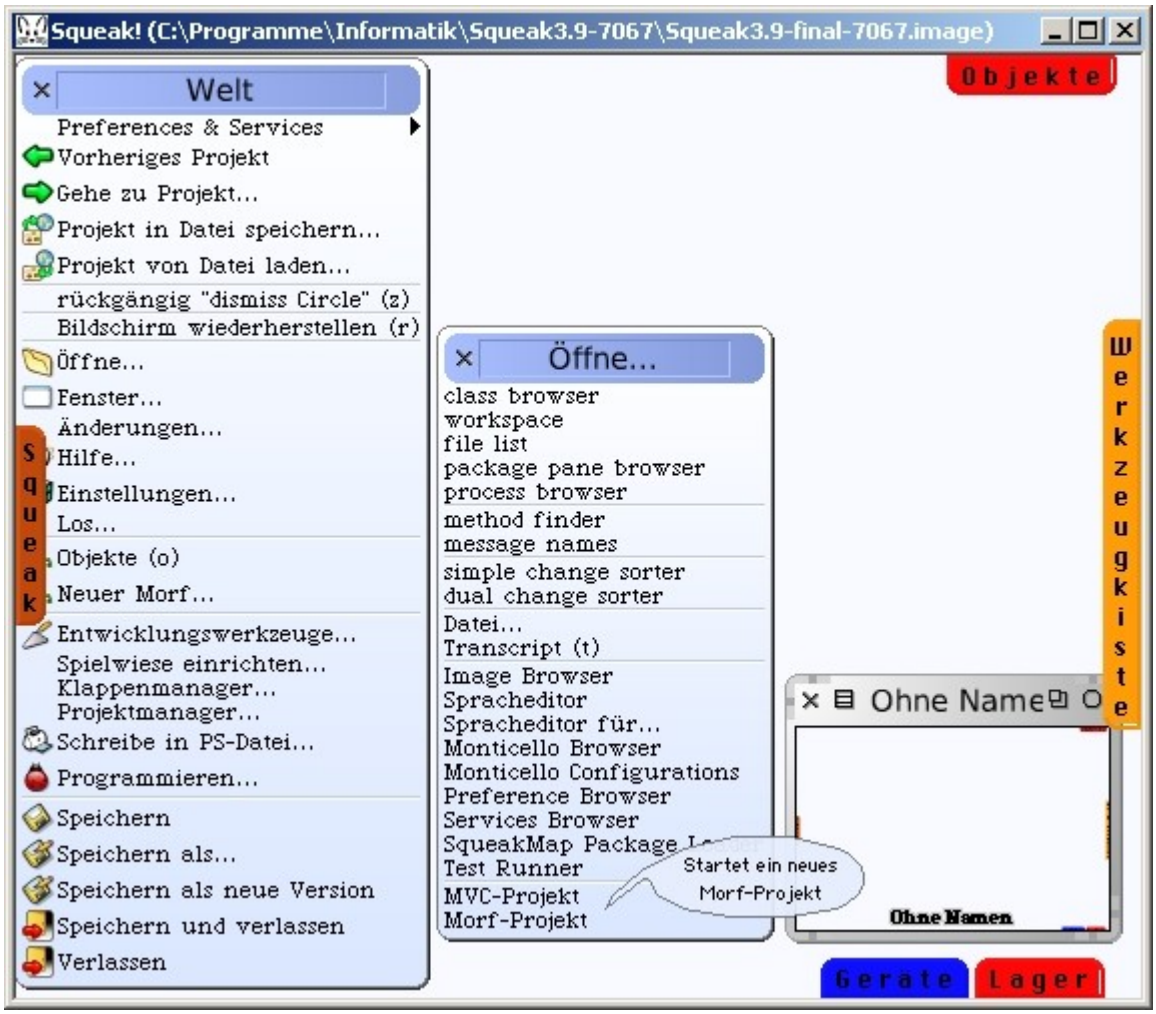


Nach Einrichtung der gewünschten Sprache, speichert man das Image unter einem neuen Namen. Somit kann zu jeder Zeit auf das Originalimage wieder zurückgegriffen werden.



Projekt Rechner anlegen

Im ersten Projekt soll ein Rechner mit den vier Grundrechenarten angelegt werden. Mit einem Linksklick öffnet sich das Menü *Welt*. Wähle den Menüeintrag *Öffne...* und dann den Menüpunkt *Morf-Projekt*



Es erscheint ein Rechteck mit dem Titel *Ohne Namen*. Klicke auf den unteren Titel und trage einen neuen Titel ein, z.B. *Taschenrechner.* Klickt man dann in die Mitte des Rechtecks mit der linken Maustaste, dann befinden wir uns in der Projektoberfläche *Taschenrechner.* In diesem Fenster soll nun der Taschenrechner realisiert werden.

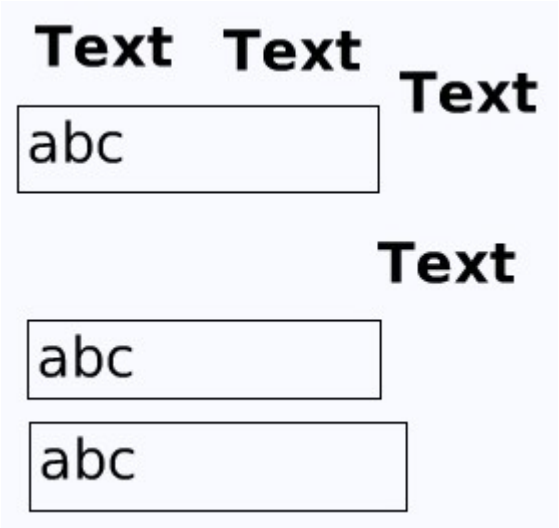
Der Taschenrechner soll über folgende Objekte verfügen:

- Zwei Textfelder für die Operanden,
- ein Textfeld für die Ausgabe,
- vier Schaltflächen für die Tasten +, -, * und /,
- eine weitere Schaltfläche für die Löschtaste,
- ein Label für die Überschrift und
- drei Label für die Textfelder der beiden Operanden und für die Ausgabe.

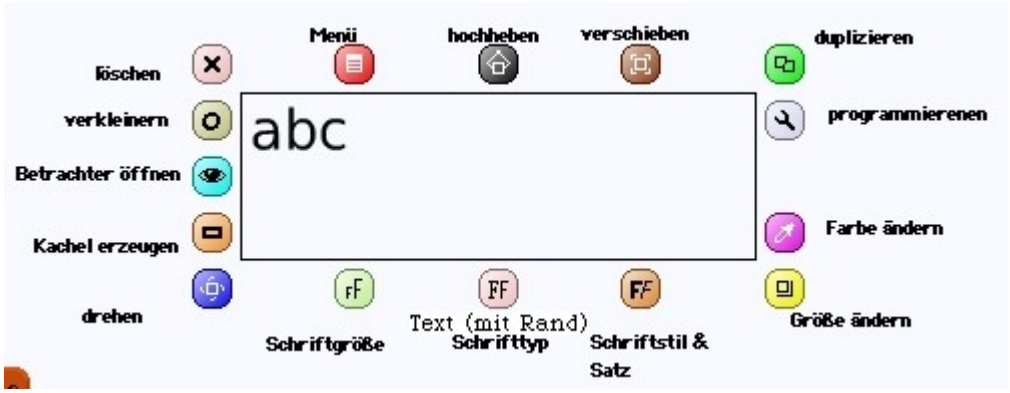
Objekte anlegen

Am Rand der Fensterfläche befinden sich mehrere Klappen. Mit Linksklick auf die Klappe *Objekte* gelangt man zur Objektauswahl.

Klicke auf die Kategorie *Text*. Ziehe nacheinander drei Textfelder mit Rand und vier Textobjekte ohne Rand auf die Fensterfläche.



Die Anordnung und Größeneinstellung der Objekte wird nun jeweils mit Hilfe eines *Halos* durchgeführt. Man erhält ein Halo zu einem Objekt durch Mittelklick oder durch ALT+Linksklick. Die Bedeutung der *Smarties* sind in der Abbildung beschrieben.

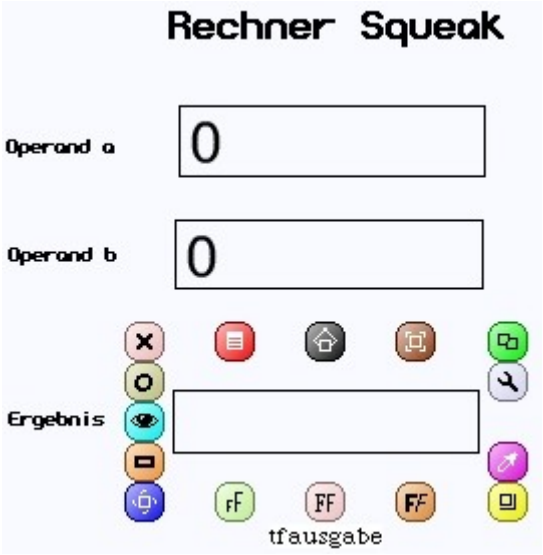


Die Anordnung, die Größenänderung und die Schrifteinstellung kann jederzeit geändert werden. Die Bezeichnungen der Objekte kann durch Überschreiben der voreingestellten Bezeichnungen unterhalb

der Objekte geändert werden. Wähle dazu aussagekräftige Namen, z.B.:

- *lbueberschrift, lboperanda, lboperandb* und *lbausgabe*; *lb* steht für Label.
- *tfoperanda, tfoperandb* und *tfausgabe*; *tf* bedeutet Textfeld.

Die Inhalte der Objekte sind aus der Abbildung zu ersehen, dabei wurde der Inhalt von *tfausgabe* durch Drücken der Leertaste erzeugt

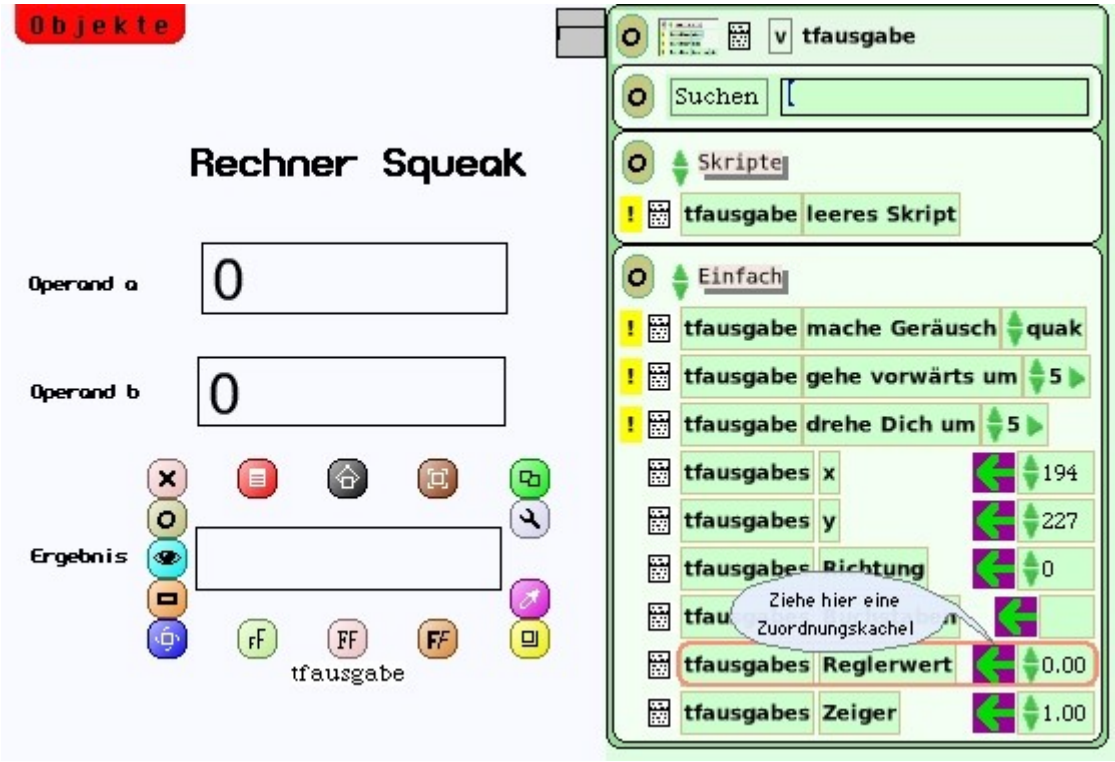


Die vier Schaltflächen +, -, *, / und *Alles löschen* werden nach der Fertigstellung der entsprechenden Methoden der Rechenoperationen und der Löschmethode erzeugt.

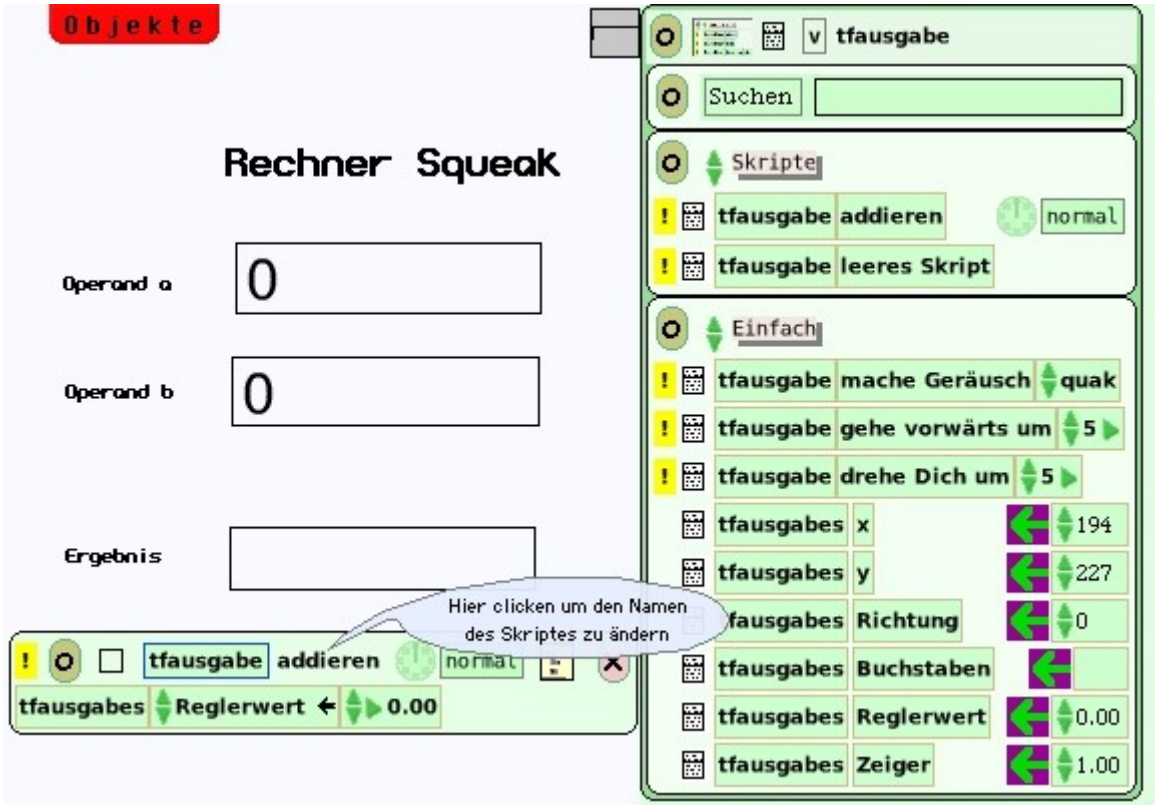
Methode addieren

Die Methode addieren verknüpft die Inhalte der Textfelder *tfoperand*a und *tfoperandb*. Das Ergebnis der Verknüpfung ist der Inhalt des Textfeldes *tfausgabe*. Die Zuordnung wird in Squeak durch := ausgedrückt.

Wir öffnen den Betrachter des Textfeldes *tfausgabe* und ziehen am Zuordnungspfeil des Reglerwertes eine Skriptkachel auf eine freie Fläche des Fensters. Achte darauf, dass die ganze Skriptkachel auf die Oberfläche gelegt wird. Ist nur der erste Teil der Befehlskachel auf der Fläche, so kann man diese Löschen, in dem man ein Halo der Kachel erzeugt und den Knopf für *Löschen x* drückt.



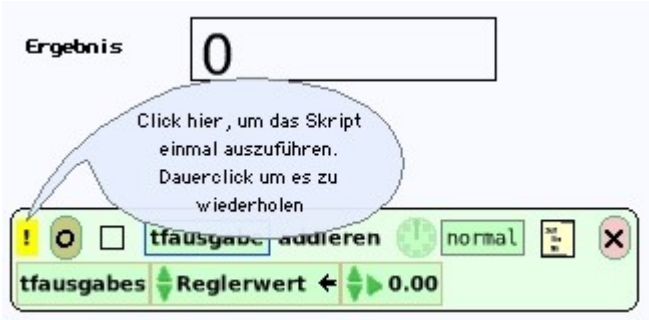
Der Name des Skriptes wird in *addieren* geändert. Klicke dazu auf den Namen Skript1 und überschreibe den Text und schließe die Eingabe mit der Eingabetaste ab. Das Skript *addieren* erscheint im Betrachter unter der Liste der Skripte. Durch Linksklick auf das Quadrat des Betrachters wird er geschlossen und auch geöffnet.



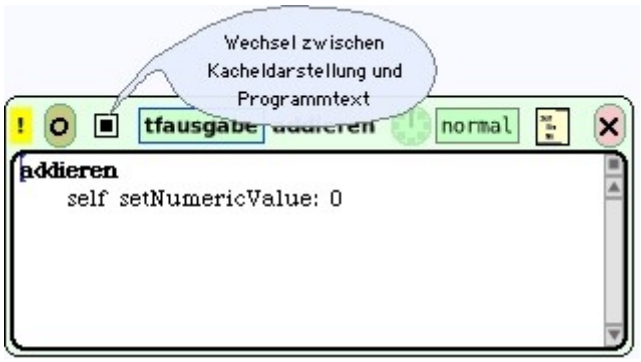
Fährt man mit der Maus über die Zeichen in der Zeile der Befehlskachel, so erscheint jeweils eine Hilfe-Blase. Mit dem x kann man das Skript löschen. Es kann dann nicht mehr wieder hergestellt

werden. Mit dem zweiten Symbol von links kann man das Skript unsichtbar machen. Im Betrachter bleibt es erhalten und kann von dort aus wieder auf die Fensterfläche gezogen werden.

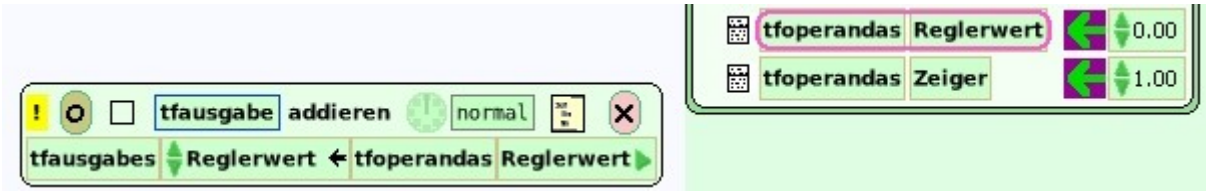
Der Inhalt von *tfausgabe* ist im Augenblick noch das Leerzeichen. Klickt man auf das Ausrufezeichen, so wird das Skript einmal ausgeführt und *tfausgabe* erhält den Wert 0.



Das dritte Symbol von links, das Quadrat, erlaubt das Umschalten von der Kachelldarstellung zum Programmtext. Wir bleiben vorläufig bei der Kacheldarstellung. Ein zweiter Klick auf das Quadrat bringt uns zur Kacheldarstellung zurück.

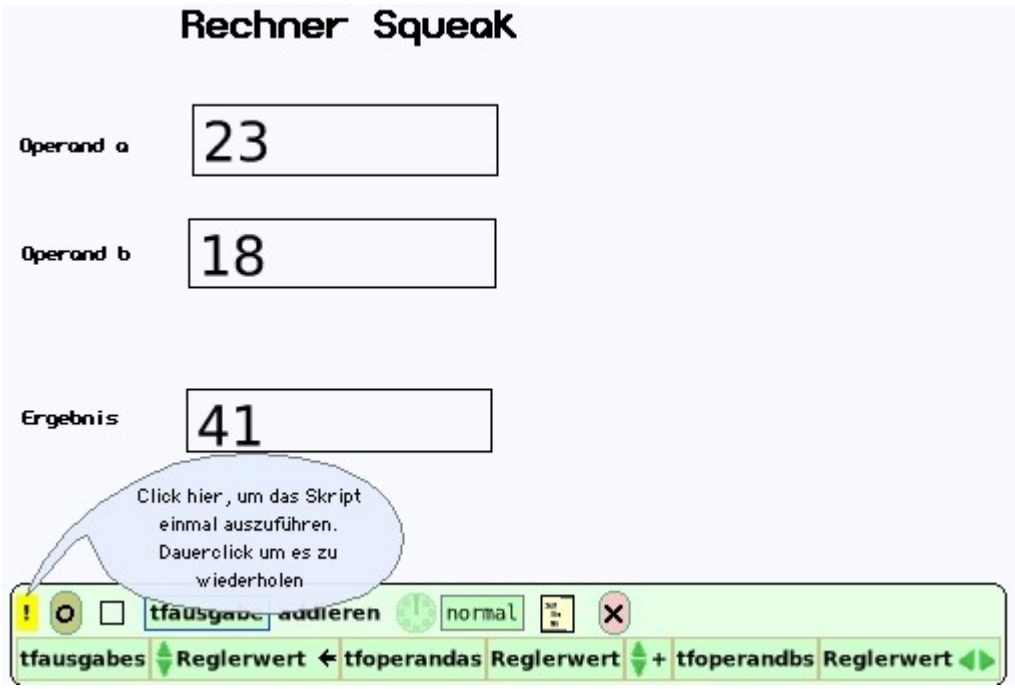


Die Summe der beiden Operanden soll nun dem Reglerwert von *tfausgabe* zugeordnet werden. Die Werte der Operanden entnehmen wir aus den zugehörigen Betrachtern der Textfelder. Öffne den Betrachter von *tfoperanda* und ziehe nur den Reglerwert auf die Zahl 0.00 in der Befehlskachel. Nun wird dem Reglerwert von *tfausgabe* der Reglerwert von *operanda* zugeordnet.

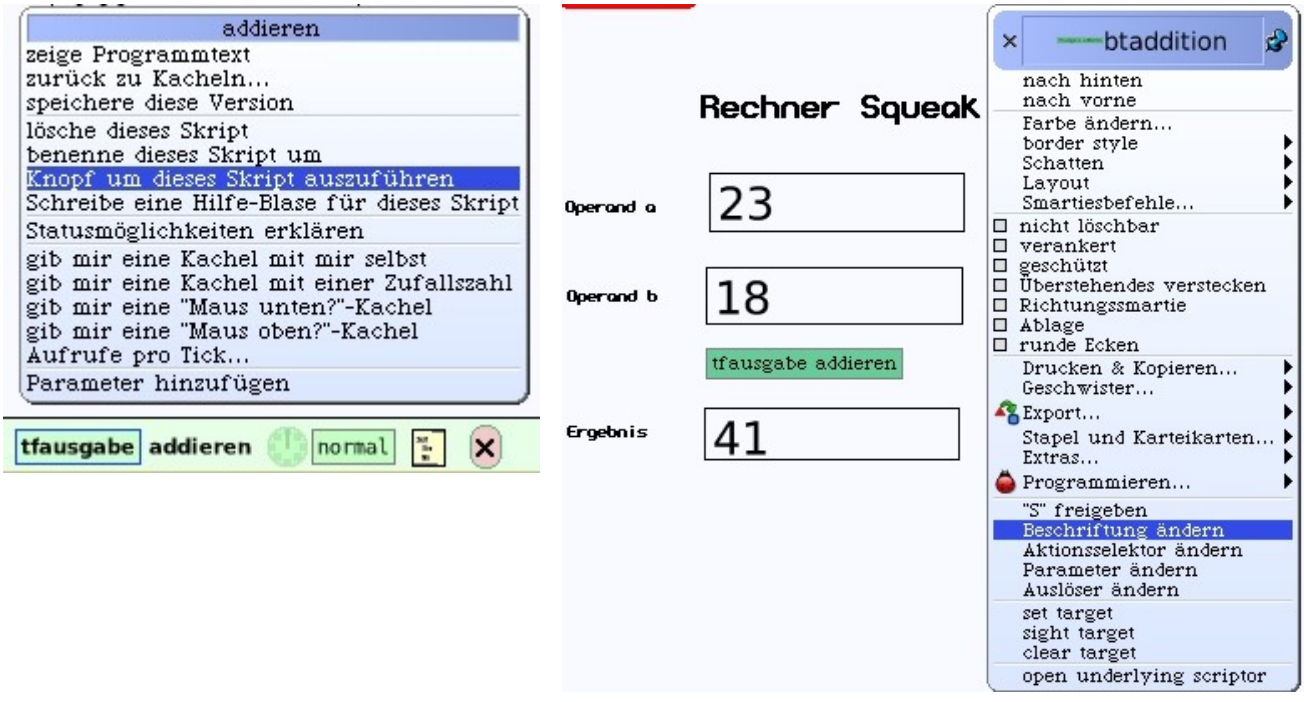


Es soll aber die Summe der beiden Operanden dem Reglerwert von *tfausgabe* zugewiesen werden. Dazu klicken wir in der Zuordnungskachel auf das nach rechts weisende Pfeilsymbol nach dem Eintrag Reglerwert. Zum Reglerwert wird nun 1 addiert. Nun öffnen wir den Betrachter des zweiten Operanden *tfoperandb* und ziehen nur den Reglerwert auf die Zahl 1 in der Befehlskachel. Die Zahl 1 wird durch den Eintrag *tfoperandbs Reglerwert* ersetzt.

Damit ist die Zuordnung für die Addition fertig und die Summe der Inhalte der beiden Summanden in den Textfelder wird im Ausgabetextfeld angezeigt.

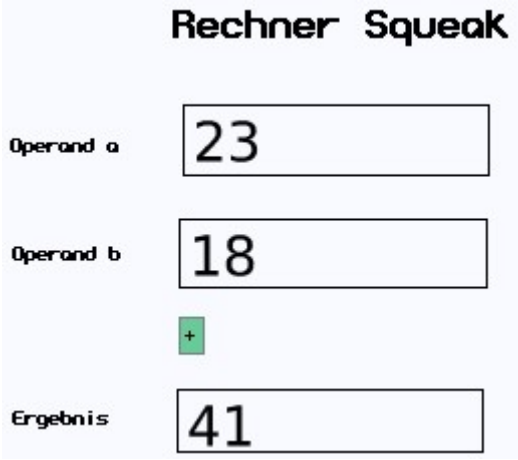


Nun fügen wir die Schaltfläche *btaddition* für die Addition hinzu. Wir klicken auf den Namen *tfausgabe* in der obersten Zeile der Skriptkachel. Es öffnet sich ein Menü und klicken auf den Eintrag *Knopf um dieses Skript auszuführen*. Wir ziehenden Knopf auf eine freie Fensterfläche. Wir ändern den Namen der Schaltfläche und wählen aus dem Halo das rote Smartie um das Menü für die Schaltfläche *btaddition* aufzurufen. Wir wählen den Menüpunkt *Beschriftung* ändern.



Es öffnet sich ein Fenster und überschreiben den Text mit dem Pluszeichen. Die Größe der Schaltfläche wird automatisch nach dem Klick auf die Schaltfläche mit der Aufschrift *OKs* angepasst. Das *s* steht für die zweite Möglichkeit die Änderung des Textes zu akzeptieren, in dem man die Tastenkombination ALT+s ausführt. Damit ist die Addition fertiggestellt. Die Skriptkachel haben wir

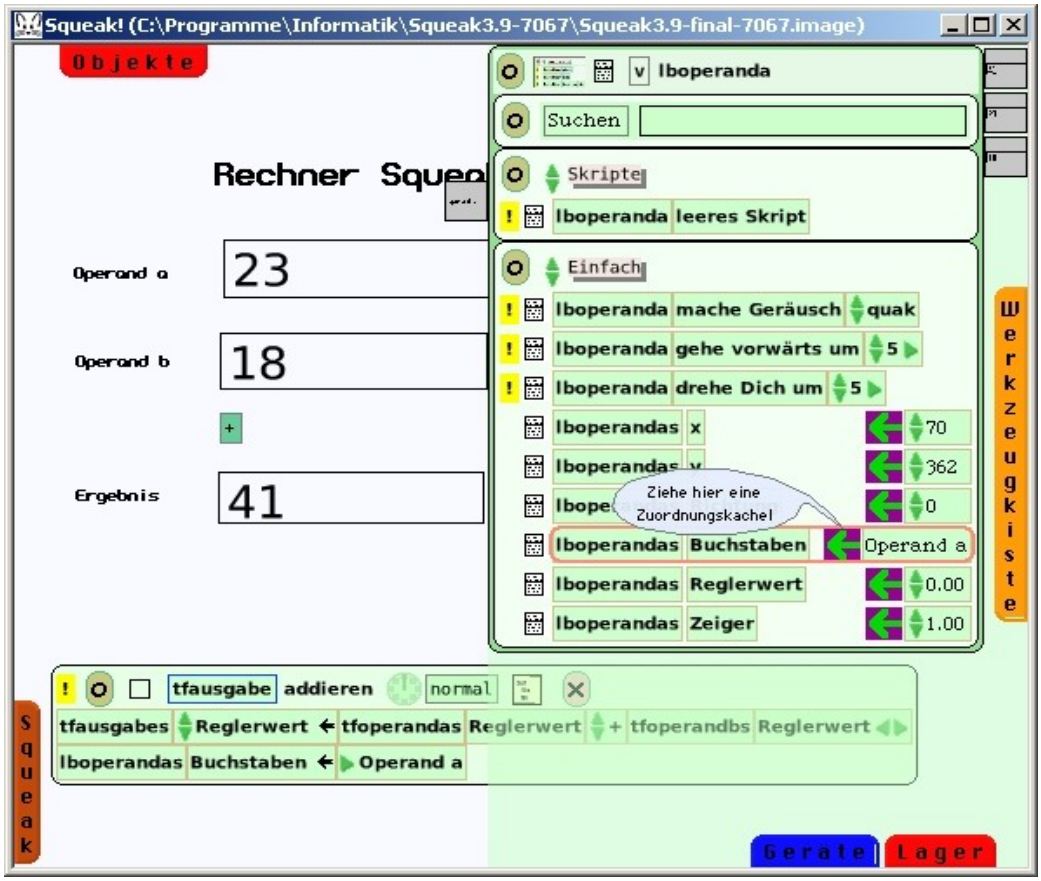
mit dem Kreissymbol unsichtbar gemacht und befindet sich im Betrachter von *tfausgabe*.



Additionsskript erweitern

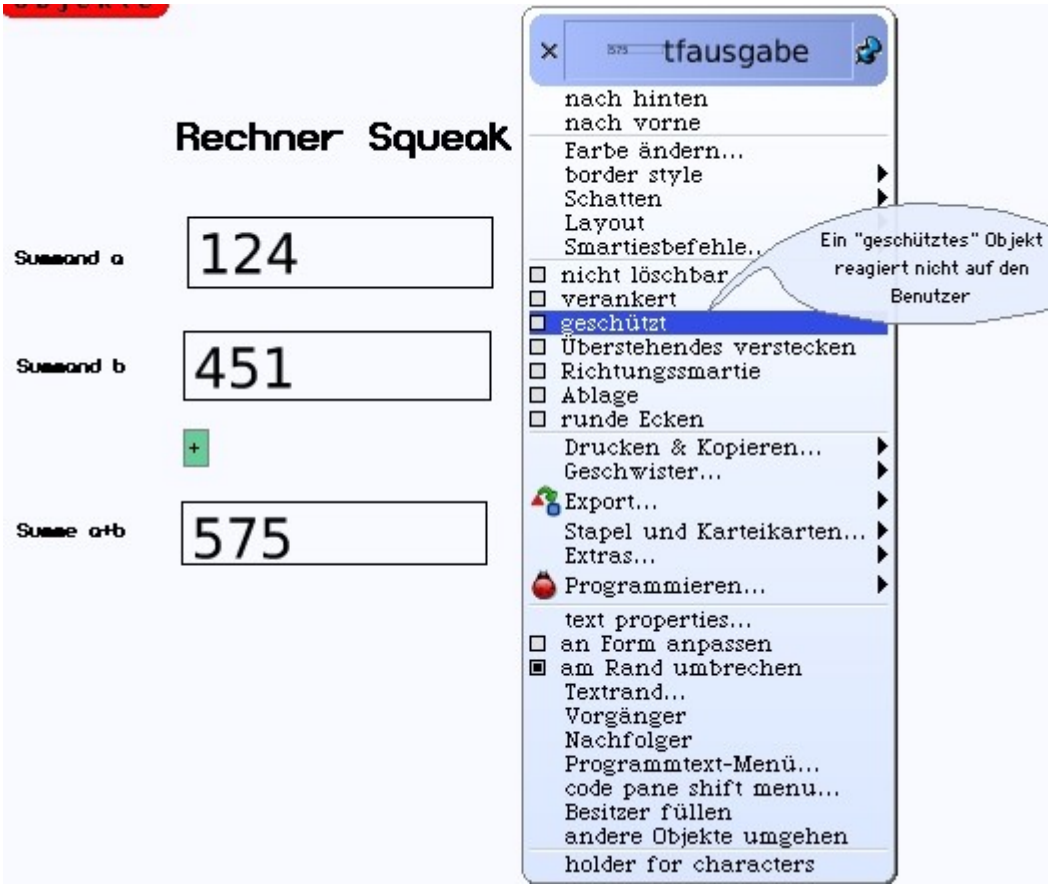
Wir werden nun das Additionsskript erweitern um die Führungstexte (Labels) der Addition anzupassen. Dabei sollen die Beschriftungen von *Operand a* in *Summand a* und entsprechend *Operand b* in *Summand b* und *Ergebnis* in *Summe a + b* bei der Ausführung der Addition geändert werden.

Wir ziehen aus dem Betrachter von *tfausgabe* das Skript *addieren*. Dann öffnen wir den Betrachter von *lboperanda* und ziehen die Zuordnungskachel für die Buchstaben in das Skript *addieren* so, dass die Zuordnungskachel unterhalb der ersten Befehlskachel Platz findet.



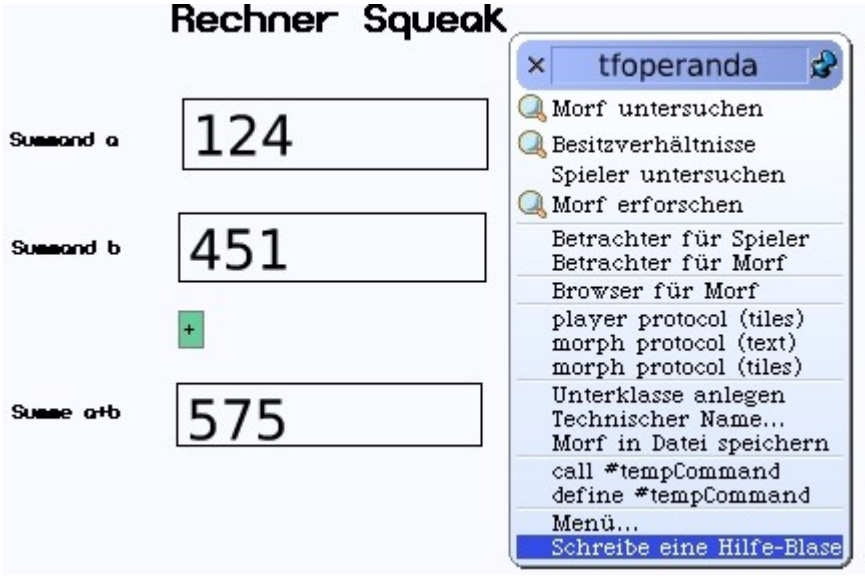
Mit Linksklick auf die Zeichenkette *Operand a* am Ende der zweiten Befehlskachel wird die Schrift rot auf blauem Hintergrund. Wir ersetzen die Zeichenkette durch die *Summand a* und schließen die Eingabe mit der Eingabetaste ab. Mit den Betrachtern für *lboperandb* und *lbergebnis* führen wir die entsprechende Schritte durch.

Wir machen das Skript *addieren* unsichtbar und führen die Addition für zwei Zahlen aus. Damit man in das Textfenster *tfausgabe* nicht eine Zahl schreiben kann, rufen wir über das Halo des Textfensters das Menü auf und wählen die Option geschützt.

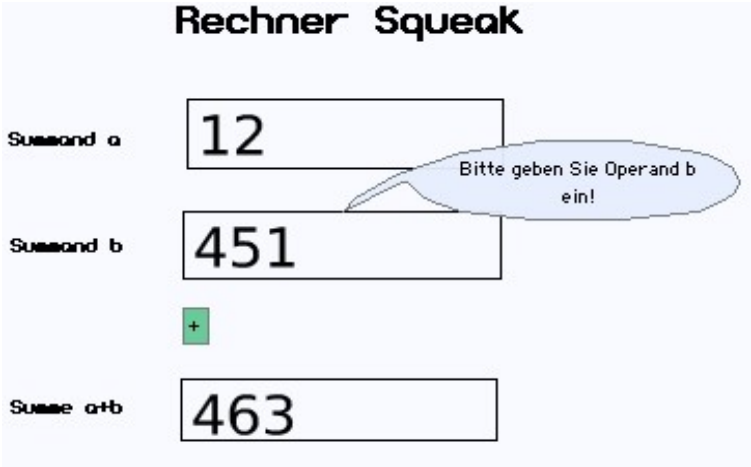


Nun kann man den Cursor nicht mehr in das Textfenster hineinsetzen. Es soll noch je eine Hilfe-Blase für die beiden Textfelder der Operanden erzeugt werden. Dazu wählen wir aus dem Halo das Symbol mit dem Schlüssel aus *Programmieren*. Aus dem Menü wählen wir den letzten Eintrag *Schreibe eine Hilfe-Blase*.

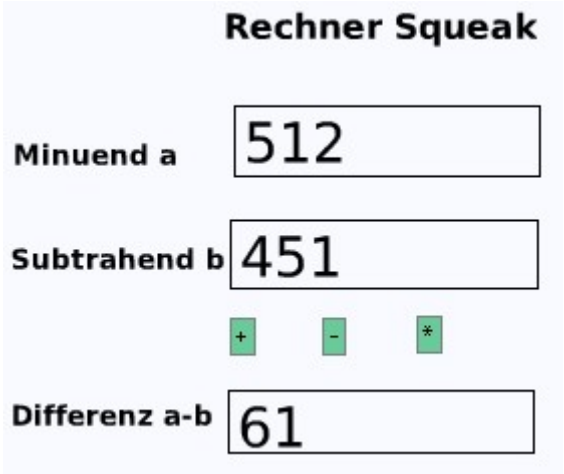
Taschenrechner

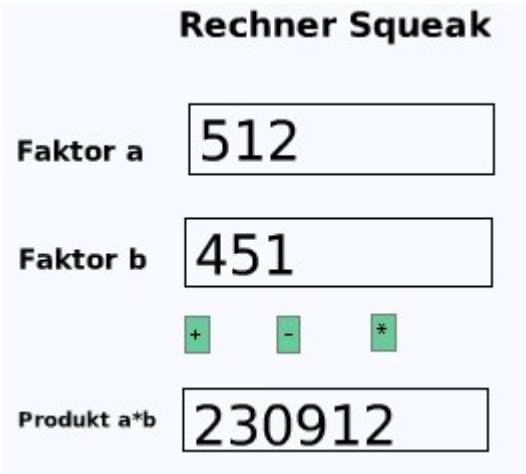


In das Textfenster schreiben wir z.B. *Bitte geben Sie Summand a ein* und schließen die Eingabe mit der Tastenkombination ALT-s ab. Will man den Text ändern, so ruft man über das Halo noch einmal den entsprechenden Menüpunkt auf und nimmt die Änderung vor. Ebenso verfassen wir einen Hilfetext für den zweiten Summanden.



Die Methoden für die Differenz und die Multiplikation können nun analog erstellt werden. Das Operationszeichen wird mit Hilfe des nach oben gerichteten Pfeiles in der Befehlskachel geändert.



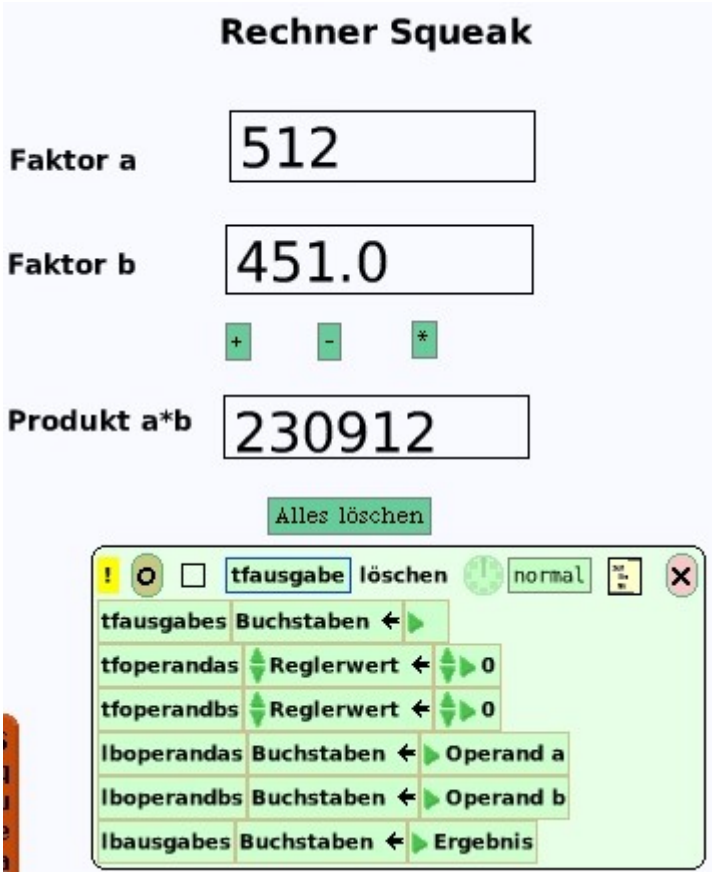


Methode löschen

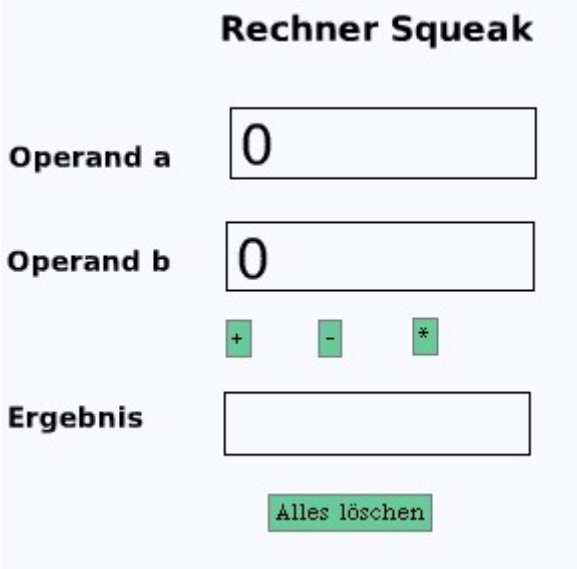
Die Methode *löschen* soll folgende Veränderungen am Rechner Squeak vornehmen:

- Die Inhalte der Eingabetextfelder werden auf 0 gesetzt,
- das Ausgabetextfeld wird gelöscht und
- die Führungstexte (Labels) werden auf die ursprünglichen Inhalte zurückgesetzt.

Wir ziehen dazu im Betrachter *tfausgabe* die Befehlskachel mit der Aufschrift *tfausgabes leeres Skript* auf eine freie Stelle. Wir ändern die Bezeichnung Skript1 zu *löschen* und schließen die Änderung mit der Eingabetaste ab. Ziehe die Zuordungskachel für die Buchstaben in das Skript. Klicke auf die Zahl und überschreibe sie, in dem Du zweimal die Leertaste drückst. Ziehe dann aus den entsprechenden Betrachtern die Zuordnungskacheln für die Reglerwerte der Operanden und überschreibe sie mit 0. Ändere jetzt noch die Führungstexte entsprechend.



Erstelle noch einen Knopf um die Methode *löschen* aufzurufen.

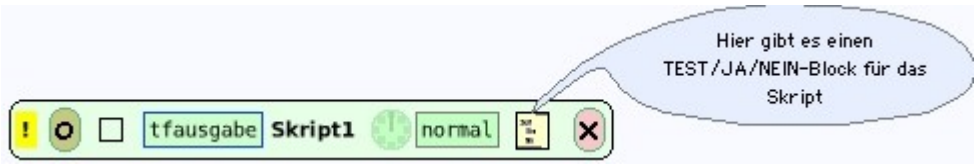


Methode dividieren

Bei der Division ist darauf zu achten, dass der Divisor von Null verschieden ist. Sind beim Quotienten *a/b* *a* und *b* gleichzeitig Null, so liegt ein unbestimmter Ausdruck vor. Jede Zahl könnte Lösung des Quotienten sein. Ist aber *a* von Null verschieden und *b* gleich Null, so ist der Quotient nicht definiert. Die Umkehrung ist dann nämlich nicht möglich. Ist *b* von Null verschieden, dann ist in jedem Fall der Quotient definiert. Wir müssen also eine Fallunterscheidung vornehmen:

```
Wenn b = 0
         dann wenn a = 0
              dann Ausgabe unbestimmter Ausdruck
              sonst Ausgabe nicht definiert
         sonst Ausgabe a/b.
```

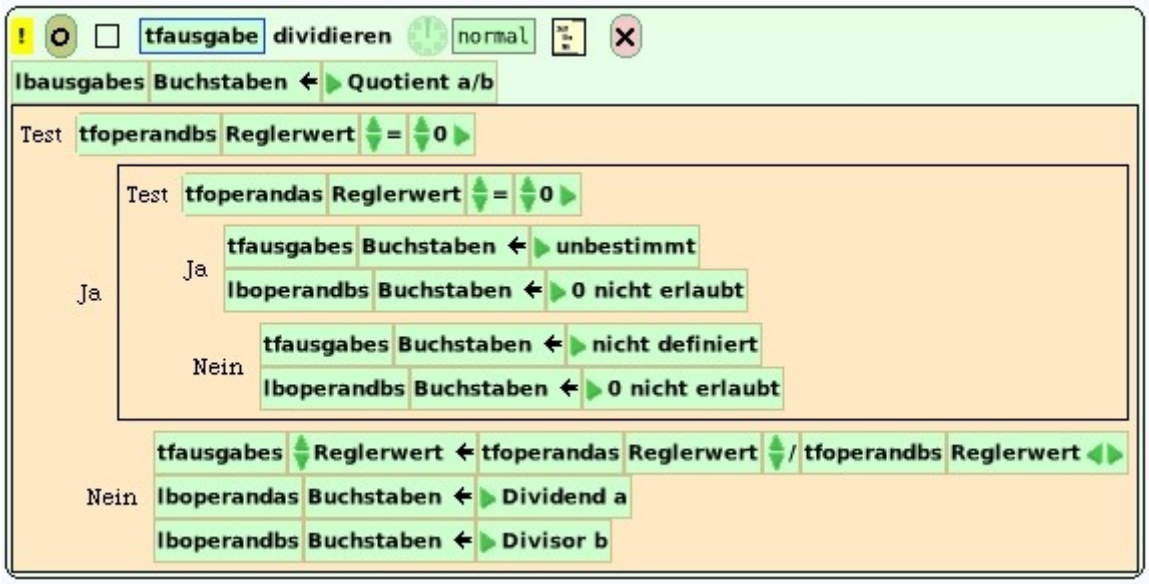
Aus dem Betrachter zu *tfausgabe* ziehen wir die leere Skriptkachel auf eine freie Stelle des Fensters.



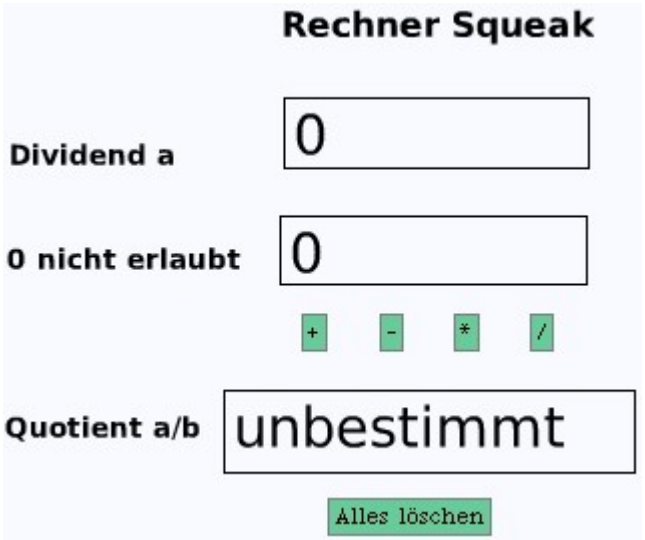
Links neben dem Kreuzsymbol finden wir ein gelbes Rechteck. Wir klicken darauf und ziehen eine Testkachel in das leere Skript und nennen es dividieren. Wir ziehen noch eine zweite Testkachel in das Skript, so dass die obige Struktur der Fallunterscheidungen schon sichtbar wird.

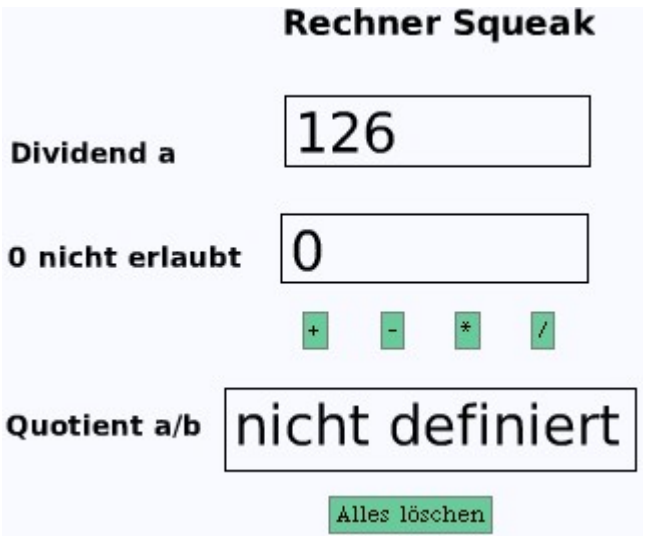
Nun müssen noch die entprechenden Anweisungen eingetragen werden. Zuerst tragen wir die booleschen Ausdrücke b = 0 und a = 0 dort ein, wo die jeweilige Überprüfung auf den Wahrheitswert der Aussage vorgenommen wird: beim Test. Achte beim Erstellen der Struktur darauf, dass der richtige Viewer (Betrachter) und nur die Reglerwerte ausgewählt werden.

Es fehlen jetzt nur noch die entprechenden Anweisungen für das Textfeld der Ausgabe. In den ersten beiden Fällen wollen wir Text (unbestimmt bzw. nicht definiert) in die Ausgabe schreiben, im letzten Fall soll der Wert des Quotienten, also ein Reglerwert, eingetragen werden. Wenn eine zweite Befehlskachel in den Testblock gezogen wird, wird neuer Platz für die Kachel direkt angezeigt.Die Labels (Führungstexte) sollten auch noch angepasst werden. Vor den JA/NEIN-Block setzen wir eine Befehlskachel für *lbausgabe*.

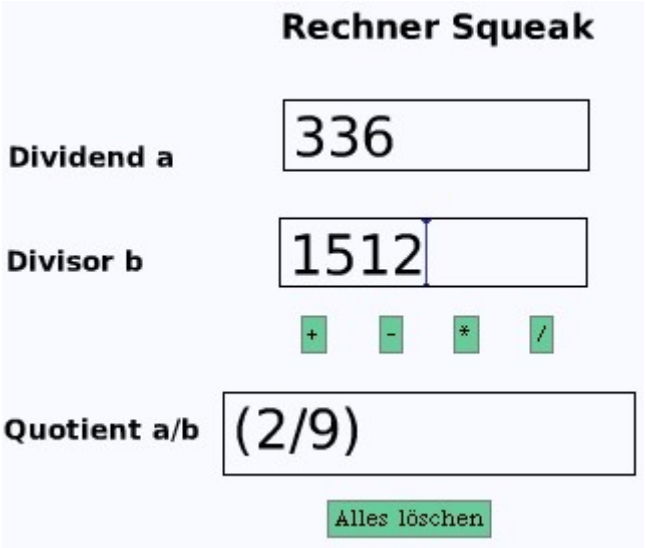


Mit der Erstellung des Knopfes zum Ausführen der Methode *dividieren* ist der Rechner für den Einsatz der Grundrechenarten abgeschlossen. Die Objekte wurden zum endgültigen Test noch ein wenig verschoben.



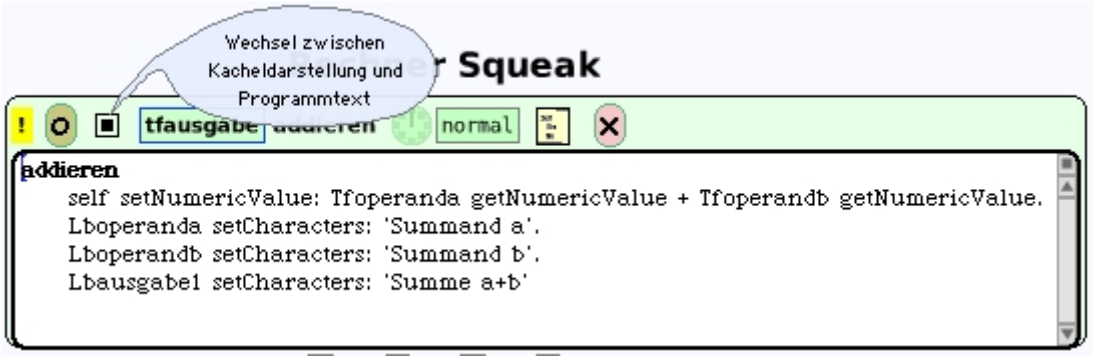


Taschenrechner



Programmtexte

Bisher haben wir die grafische Programmierung zur Erstellung der grafischen Oberfläche (GUI: graphical user interface) und Programmierung des Rechners mit Befehlskacheln eingesetzt. Wir wollen nun die entsprechenden Befehle der Programmiersprache Smalltalk uns genauer anschauen. Dazu schauen wir uns die Methode *addieren* in der Ansicht *Programmtext* an.



Wir erkennen, dass im Programmtext die verwendeten Objekte mit ihren Namen auftreten:

- self (damit ist das Objekt *tfausgabe* gemeint),
- Tfoperanda,
- Tfoperandb,
- Lboperanda,
- Lboperandb und
- Lbausgabe1

Die Objekttypen **Text (mit Rand)** (Textfelder) und **Text** (Label) stellt Squeak zur Verfügung. Die Eigenschaften der Objekte sind in Klassen festgelegt worden. Zusätzlich wurde bei der Definition der Klassen auch festgelegt, wie man die Eigenschaften **auslesen** und **verändern** kann. Dies erfolgt durch eine entsprechende **Nachricht**. Ein Objekt verfügt für jede Nachricht, die es versteht, über eine so genannte **Methode**, in der die Reaktion des Objekts auf den Empfang der entsprechenden Nachrichten festgelegt (programmiert) ist.

Die Grundform des Nachrichtenausdrucks in Smalltalk lautet also:

```
<objekt> <nachricht>
```

Das Ergebnis ist immer ein Objekt!

Programme in der Sprache Smalltalk bestehen grundsätzlich aus **Nachrichten** die an **Objekte** gesandt werden. Das Ergebnis einer solchen Nachricht ist wieder ein Objekt. (aus Johannes Brauer, Grundkurs Smalltalk, Vieweg)

Die Methoden, die im Programmtext *addieren* auftreten, sind:

- getNumericValue
- setNumericValue:
- setCharacters:
- + Zahlobjekt

Die Nachricht *getNumericValue* wird von einem Objekt der Klasse **Text (mit Rand)** verstanden und gibt als Ergebnis die eingegebene Zahl als Objekt zurück. Diese Nachricht besitzt kein Argument; es handelt sich um eine *unäre Nachricht*.

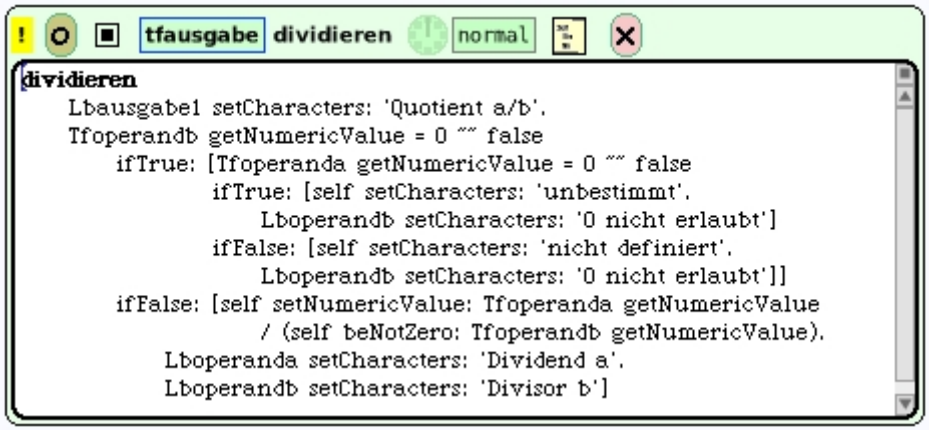
Die Nachricht *setNumericValue:* wird von einem Objekt der Klasse **Text (mit Rand)** verstanden und verändert den Inhalt eines Textfeldes. Im Argument steht nach dem Doppelpunkt die Information, die in das Textfeld eingetragen werden soll. Es handelt sich bei solchen Nachrichten um so genannte *Schlüsselwortnachrichten*.

Die Nachricht *setCharacters:* ist ebenfalls eine Schlüsselwortnachricht, die von einem Objekt der Klasse **Text** verstanden wird. Im Argument steht die Zeichenkette in Hochkommata, die den Inhalt des Textes festlegt.

Die Nachricht **+ Zahlobjekt** wird an ein Zahlenobjekt gesandt, denn die Aufrufe *tfoperanda getNumericValue* und *tfoperandb getNumericValue* geben ja Zahlenobjekte zurück. Eine solche Nachricht mit genau einem Argument heißen *binäre Nachrichten*.

Der Programmtext beginnt mit dem Namen der Methode **addieren**. Dann folgen einzelne Zeilen, die mit einem Punkt enden. Der Abschlusspunkt in der letzten Zeile kann wie hier fehlen.

Auch *ifTrue:* ist ein Schlüsselwort. Das zugehörige Argument wird in eckige Klammern gesetzt. Man nennt dies einen Block. Analysiere den Programmtext der Methode *dividieren*.



Verändert man den Programmtext im Skript, so kann die Kacheldarstellung nicht mehr eingesetzt werden. Nicht zu jedem Befehl gibt es nämlich eine Kachel. Daher werden sämtliche Änderungen, die man im Programmtext vorgenommen hat wieder rückgängig gemacht, wenn man wieder zur Kacheldarstellung zurück geht.

Wurzel, Potenzieren

Der Rechner Squeak soll nun noch zwei weitere Funktionen erhalten, die nicht mit der Kacheldarstellung programmiert werden können: Wurzelziehen und Potenzieren. Squeak stellt für beiden Operationen entsprechende Nachrichten zur Verfügung:

- Wurzel aus 5:

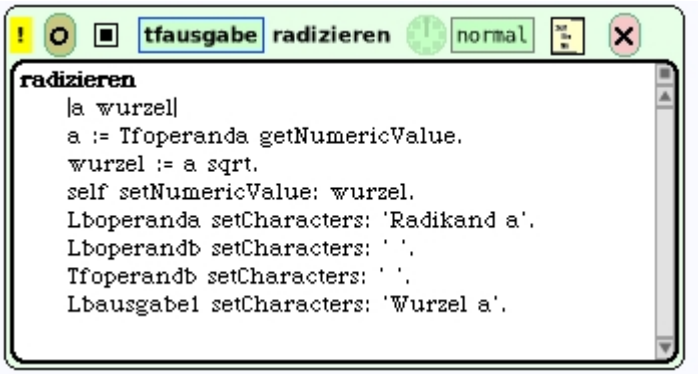
```
5 sqrt
```

- 5 hoch 3:

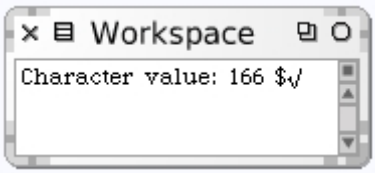
```
5 raisedTo: 3
```

Wir öffnen den Betrachter von *tfausgabe* und ziehen ein leeres Skript auf eine freie Stelle. Wir nennen das Skript *radizieren* und wechseln zur Programmtextdarstellung. Zuerst lesen wir den Inhalt des Textfeldes von *Operand a* aus und ordnen den Wert der Variablen *a* zu. Diese muss lokal zu Beginn des Programmtextes festgelegt werden. Ferner verwenden wir noch die lokale Variable *wurzel*, die den Wurzelwert aufnimmt. Die lokalen Variablen werden nach dem Methodennamen angegeben. Dabei werden sie durch zwei senkrechte Striche | | (Alt Gr + <) begrenzt.

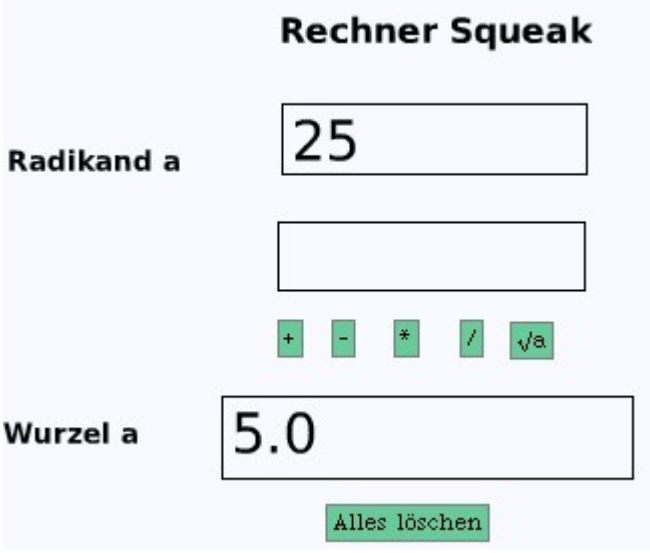
Als Zuweisungsoperator verwenden wir die Zeichenfolge :=.



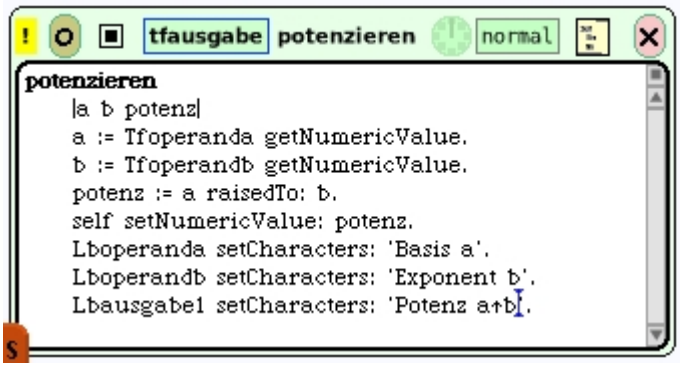
Im Programmtext sind die Führungstexte (Labels) angepasst worden. Der Programmtext wird immer vor der Ausführung mit ALT-s gesichert. Ist ein Bezeichner oder eine Methode nicht richtig geschrieben worden, so macht Squeak Vorschläge für eine entsprechende Änderung. Der Knopf soll das Wurzelzeichen enthalten. Wir kopieren das Zeichen ALT-c aus dem Workspace und fügen es mit ALT-v in das Fenster für die Beschriftung ein. Das Werkzeug Workspace findet man bei der Klappe Werkzeuge.

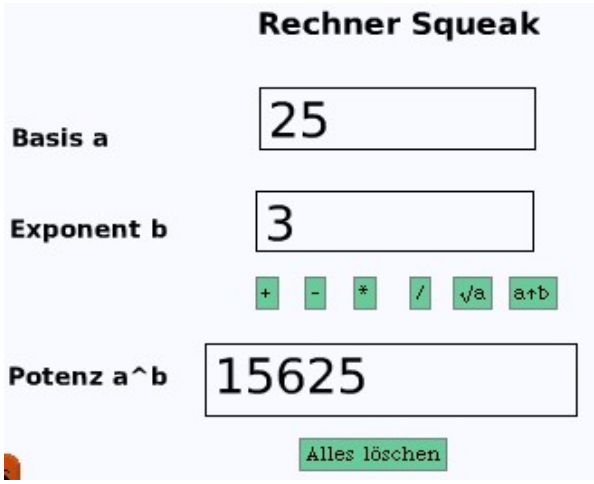


Das Sonderzeichen erhalten wir durch das abgebildete Skript im Workspace. Man Führt es nach Speicherung (mit ALT-s) mit der Tastenkombination ALT-p (Print it) aus.



Bei der Implementation des Potenzierens gehen wir entsprechend vor.

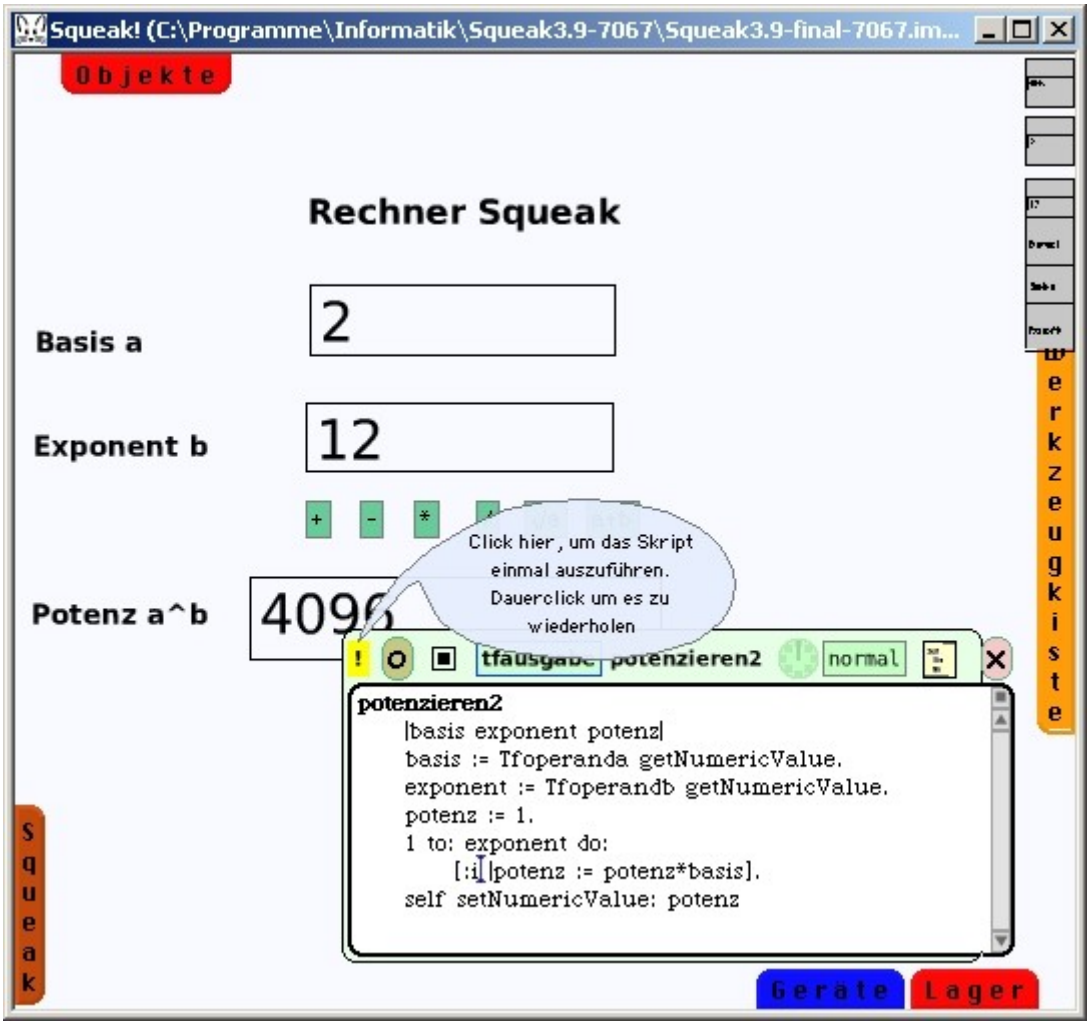




Das Potenzieren von natürlichen Basen und nicht negativen, natürlichen Exponenten soll nun mit einer eigenen Methode programmiert werden. Ist der Exponent 0, so ist das Ergebnis der Potenz 1, im anderen Fall muss die Basis so oft mit sich selbst multipliziert werden, wie der Exponent angibt. Dies realisieren wir mit einem Intervalldurchlauf. In Squeak wird diese Intervallschleife folgendermaßen benutzt:

```
n to: m do: [ :i | <Anweisungsfolge> ]
```

Die Bezeichner *n* und *m* stehen für die Intervallgrenzen des Intervalls, welches mit der Schrittweite 1 durchlaufen wird. Der Block beginnt mit der Blockvariablen *i*, die den jeweiligen aktuellen Wert beim Schleifendurchlauf aufnimmt. Für die Methode *potenzieren* ergibt sich in Squeak der obige Programmtext.



Weitere Operationen für den *Rechner Squeak* sollten nun selbstständig eingearbeitet werden können. Als neues Projekt bietet sich die Erstellung eines Bruchrechners an. Viel Freude bei der Umsetzung.