# Grafische Programmierung

## mit Ruby

Tahrin Alam
Zixiang Gu

**Was ist grafische Programmierung**

**Tk & andere GUI-Libraries**

**Einführung in Tk mit Ruby**

**Übungsaufgaben**

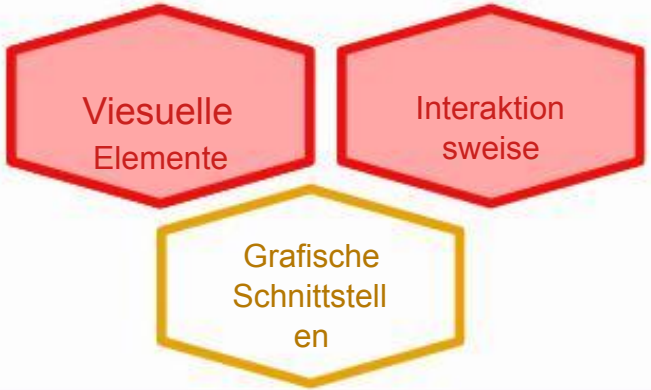
# shorturl.at/gpBZ4

https://github.com/TahrinAlam/proseminar-ruby-grafische-programmierung

# Konzept von Grafische Programmierung

**Gegenteil von Textkommando-Programmierung:**

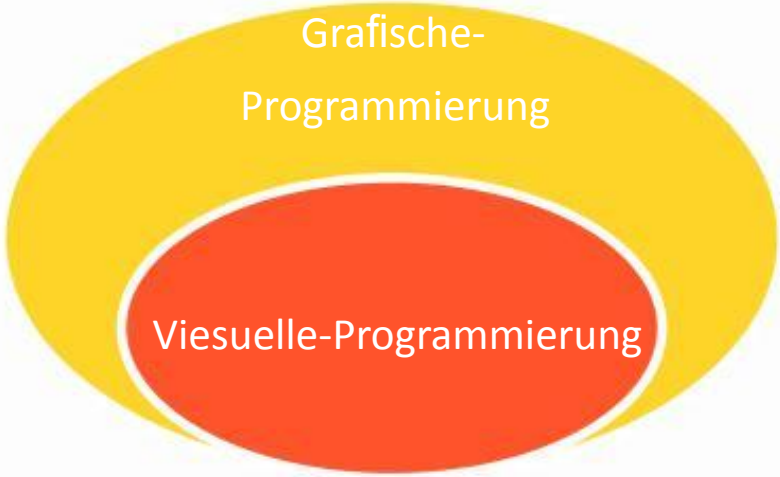
- Visuelle Elemente
- grafische Schnittstellen (GUI)
- Interaktionsweisen



# Konzept von Grafische Programmierung

**Unterschied zur visuelle-Programmierung:**

verschiedenen Methoden und Tools zur Programmierung mit grafischer Darstellung



# Grafishe Programmiersprache Beispiel

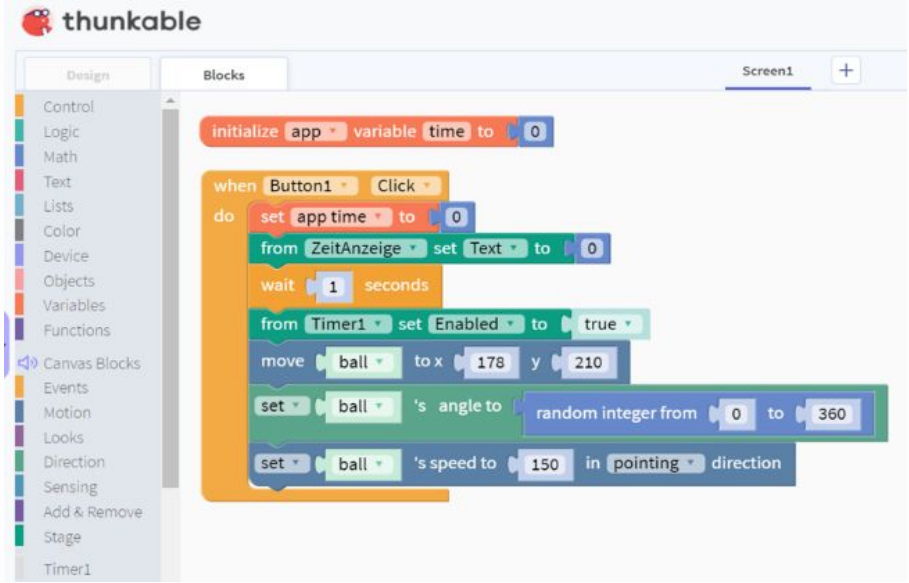
- Blockbasierte Programmiersprache

- Grafischen-Bibliothek für GUI

# Blockbasierte Programmiersprache

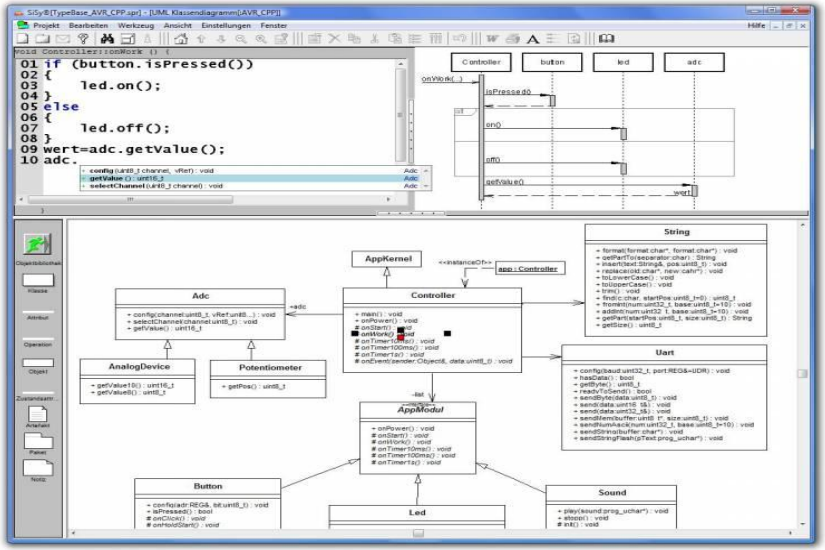
Programmierkonzepte => visuelle Blöcke

Scratch, Blockly



# Grafischen-Bibliothek für GUI

- Datenbindung
- Eventbehandlung
- Layoutverwaltung
- Stil und Design
- Zeichnen und Grafiken



# Grafischen-Bibliothek für GUI

Java: JavaFx, AWT
Python: Tkinter,PyQt
Ruby: Tk, Shoes, Glimmer

# Grafische Programmierung im Ruby

**Ruby**

Objekorientierte

Dynamische

Einfach Syntax

Intergrierte Klassenbibliotheken

# Grafische Programmierung im Ruby

Ziel: visuell ansprechende und interaktive Anwendungen aufbauen

# Grafische Programmierung im Ruby

Ziel: visuell ansprechende und interaktive Anwendungen aufbauen

# Grafische Programmierung im Ruby

Ziel: visuell ansprechende und interaktive Anwendungen aufbauen

# Welche Grafishce Toolkit?

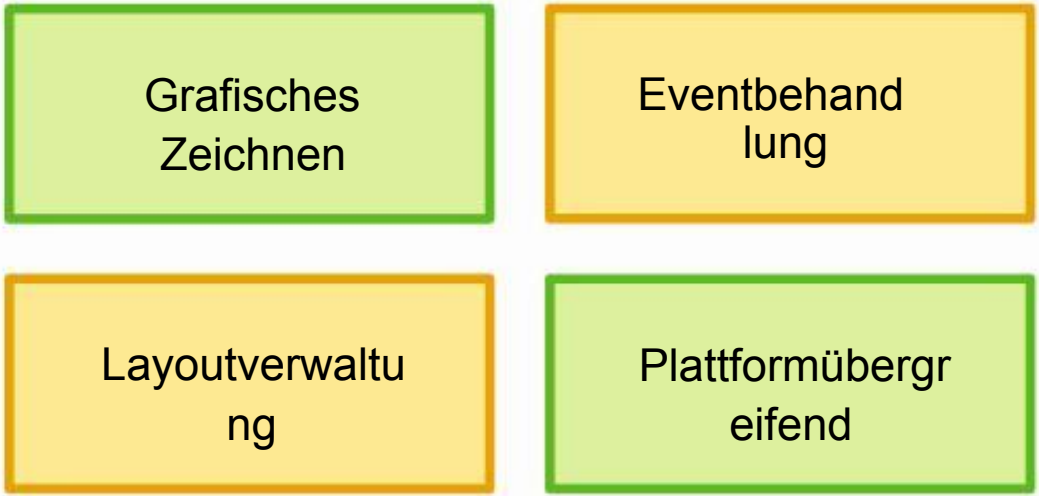
# Shoes

Jonathan Gillette 2007 (Hackety Hack)

Shoes 3 => Shoes 4

Framework und Toolkit in Ruby

# Shoes Eingenschaften

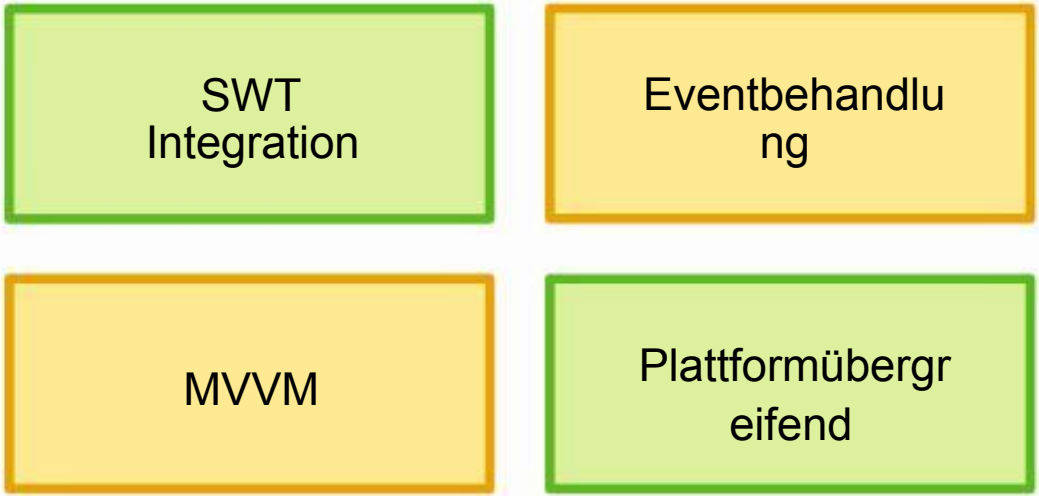


# Glimmer

Framework und Toolkit in Ruby

Jruby-Implementierung um die Anwendung auf JVM auszuführen
deklarativen DSL(Domain-Specific Language) : “SWT(Standard Widget Toolkit) XML “

# Glimmer Eingenschaften



# Tk

Die grafische Standardbenutzeroberfläche (GUI) Toolkit für Ruby

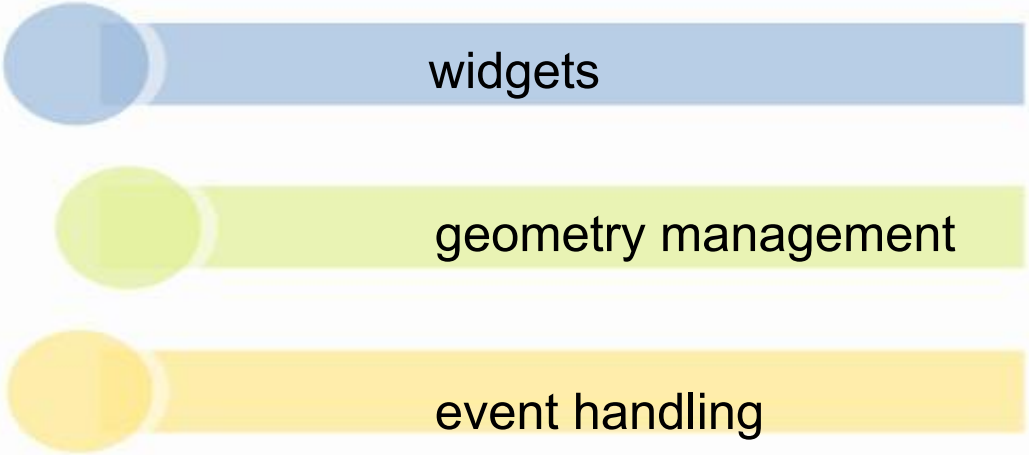
Tcl-Skriptsprache von John Ousterhout

Mit Hilfe von Paketverwaltungssystem (Gem) installieren

# TK Eingenschaften

- Datenbindung
- Eventbehandlung
- Benutzerdefinierte
- Plattformübergreifend

# Konzept von Tk



# Wieso Tk ausgewählt?

Stabilität: Tk > Shoes, Glimmer

Einfachkeit: Shoes > Tk > Glimmer

erweiterbarkeit: Glimmer > Tk > Shoes

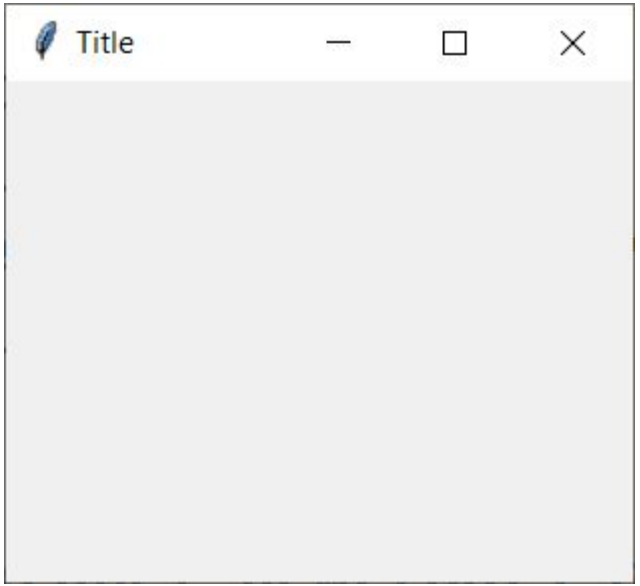
# Einführung in Tk mit Ruby

# Grundlagen

- `TkRoot` als Hauptfenster, `Tk.mainloop` zum Starten der GUI
- Widget basiert ➜ jedes angezeigte Element ist ein Widget
- Positionierung der Widgets mit `pack grid place`
- Interaktion der Widgets durch binden an Events

# Fenster erstellen

```
require 'tk'

root = TkRoot.new do
  minsize 200, 150
  maxsize 400, 500
  resizable false, true
  title 'Titel'
end

Tk.mainloop
```



# Positionierung

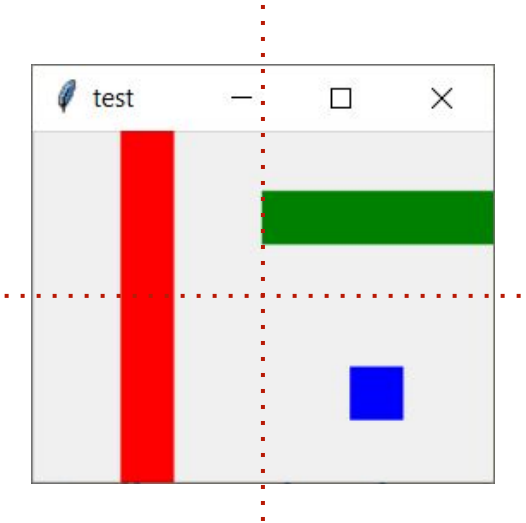
# grid

```
TkFrame.new(root) do
  background 'red'
  width 25
  height 25
  grid(row: 0,
       column: 0,
       rowspan: 2,
       sticky: 'ns')
end
```

```
TkFrame.new(root) do
  background 'green'
  width 25
  height 25
  grid(row: 0,
       column: 1,
       sticky: 'ew')
end
```

```
TkFrame.new(root) do
  background 'blue'
  width 25
  height 25
  grid(row: 1,
       column: 1)
end
```

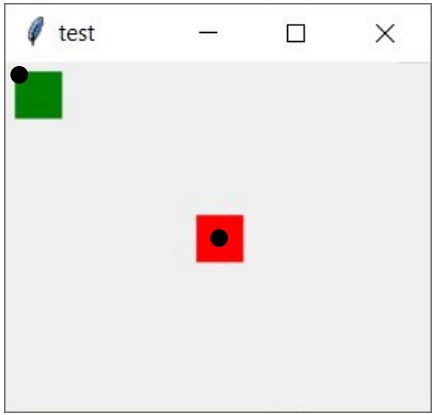
```
root.grid_columnconfigure((0..1).to_a, weight: 1)
root.grid_rowconfigure((0..1).to_a, weight: 1)
```



# place

```
TkFrame.new(root) do
  background 'red'
  width 25
  height 25
  place(relx: 0.5,
        rely: 0.5,
        anchor: 'center')
end
```

```
TkFrame.new(root) do
  background 'green'
  width 25
  height 25
  place(x: 5, y: 5)
end
```

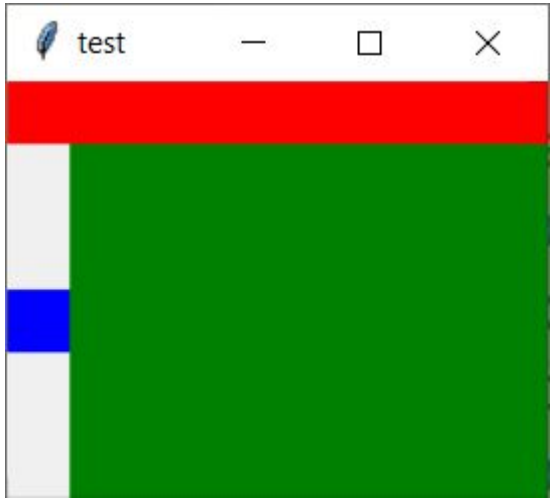


# pack

```
TkFrame.new(root) do
  background 'red'
  width 25
  height 25
  pack(side: 'top',
       fill: 'x')
end
```

```
TkFrame.new(root) do
  background 'blue'
  width 25
  height 25
  pack(side: 'left')
end
```

```
TkFrame.new(root) do
  background 'green'
  width 25
  height 25
  pack(side: 'right',
       fill: 'both',
       expand: true)
end
```



# Interaktivität

# Events

- jede Interaktion/Aktion löst Events aus
- Widgets haben vordefinierte Reaktionen auf Events (z.B. Doppelklick auf Text ➜ markieren)
- binden an Events mit `bind '<Event>' do … end`
- Erstellen und Auslösen selbstdefinierter Events möglich

| Event | Beschreibung |
|---|---|
| `<Button-X>` | Button 1 is the leftmost button, button 2 is the middle button(where available), and button 3 the rightmost button.<br><br>`<Button-1>`, `<ButtonPress-1>`, and `<1>` are all synonyms.<br><br>For mouse wheel support under Linux, use Button-4 (scroll up) and Button-5 (scroll down) |
| `<Key>` | The user pressed any key. The key is provided in the char member of the event object passed to the callback (this is an empty string for special keys). |
| `<Map>` | A widget is being mapped, that is, made visible in the application. This will happen, for example, when you call the widget's `.grid()` method. |

**Für mehr Event-Typen:**

https://tcl.tk/man/tcl8.6/TkCmd/bind.htm#M7

https://web.archive.org/web/20190512164300id_/http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/event-types.html

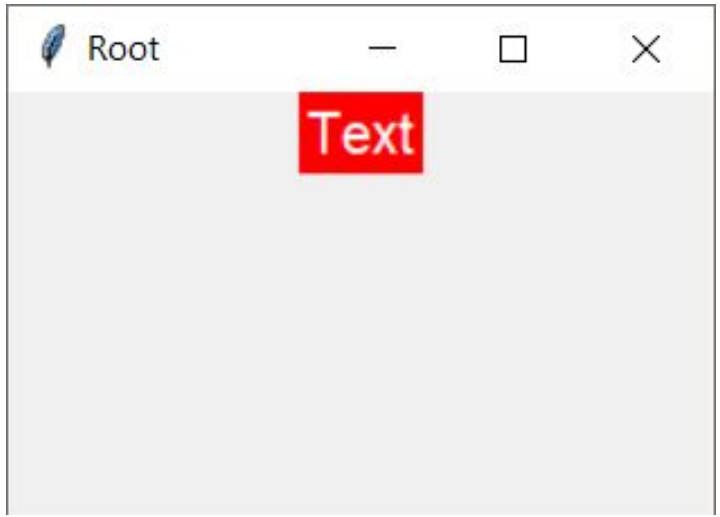
# bind Syntax

```
root = TkRoot.new do
  bind '<Escape>' do
    puts 'Bye'
    destroy
  end
end
```

# Widget-Typen

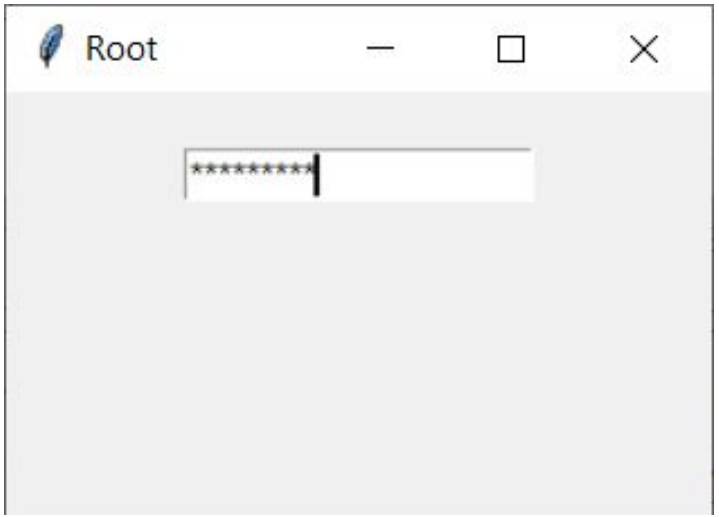
# TkLabel

```
TkLabel.new(root) do
  text 'Text'
  font '50'
  foreground 'white'
  background 'red'
  pack
end
```



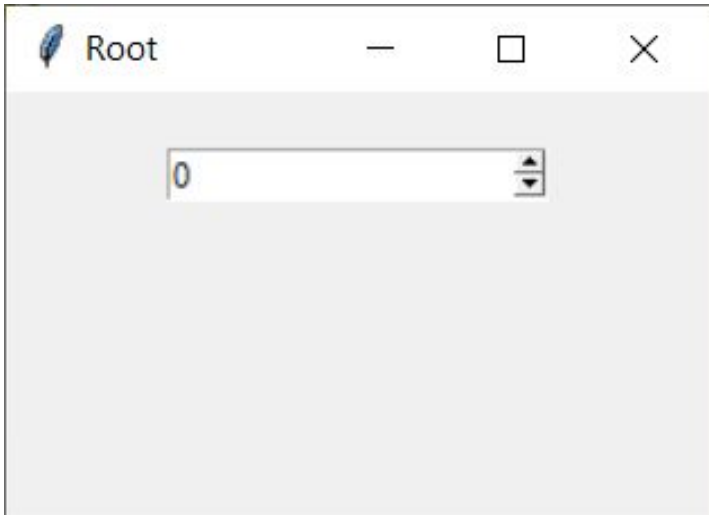
# TkEntry

```
TkEntry.new(root) do
  width 20
  show '*'
  pack(pady: 20)
end
```



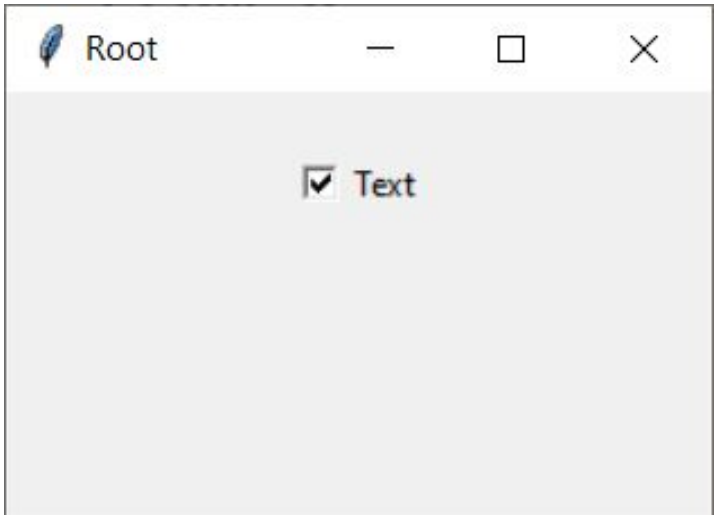
# TkSpinbox

```
TkSpinbox.new(root) do
  from 0
  to 100
  increment 1
  textvariable
  pack(pady: [20, 0])
end
```



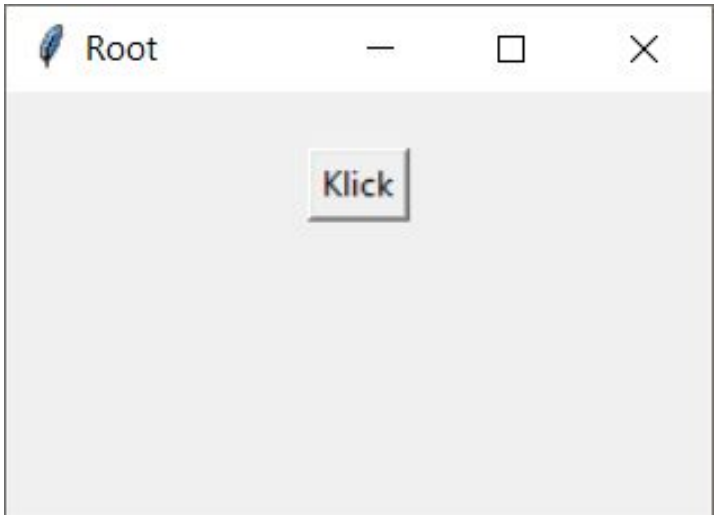
# TkCheckButton

```
TkCheckButton.new(root) do
  text 'Text'
  pack(pady: 20)
end
```



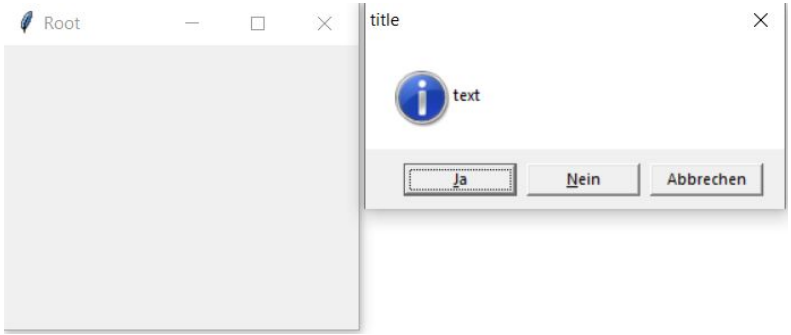
# TkButton

```
btn = TkButton.new(root) do
  text 'Klick'
  command proc {
    btn.text = 'geklickt!'
  }
  pack(pady: 20)
end
```



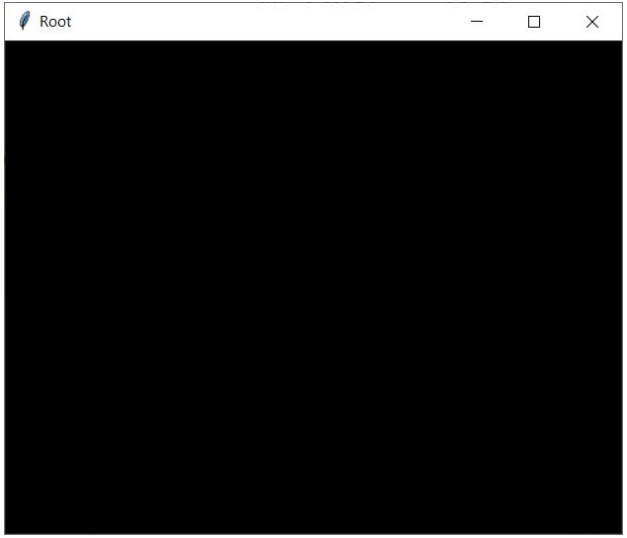
# Tk.messageBox

```
msg_box = Tk.messageBox(type: 'yesnocancel',
                        message: 'text',
                        title: 'title')
msg_box.call
```



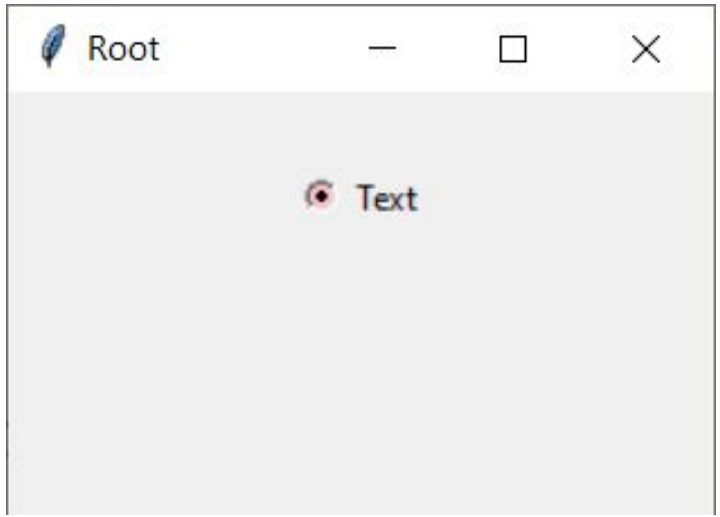
# TkFrame

```
TkFrame.new do
  relief 'sunken'
  background 'black'
  height 400
  width 500
  pack
end
```



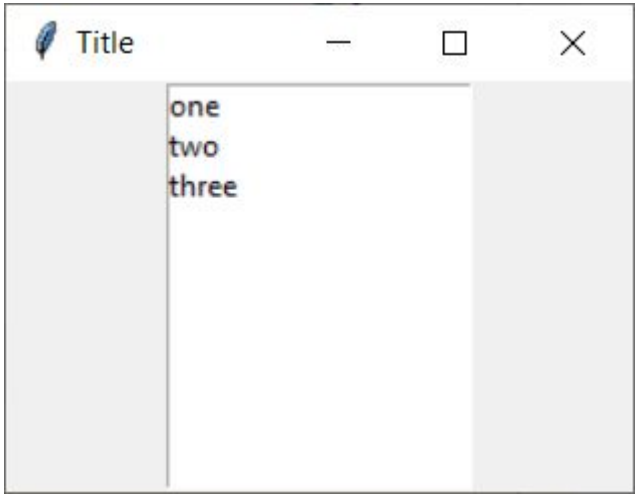
# TkRadiobutton

```
TkRadiobutton.new(root) do
  selectcolor 'pink'
  text 'Text'
  pack(pady:25)
end
```



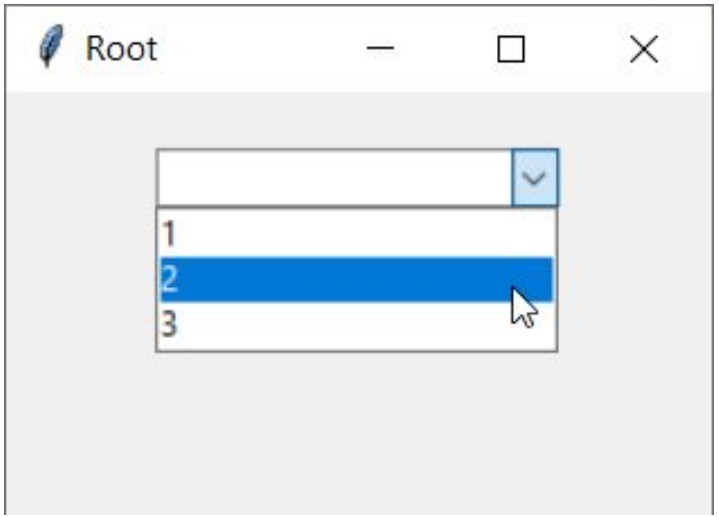
# TkListbox

```
var = TkVariable.new(%w[one two three])
TkListbox.new(root) do
  listvariable var
  pack
end
```



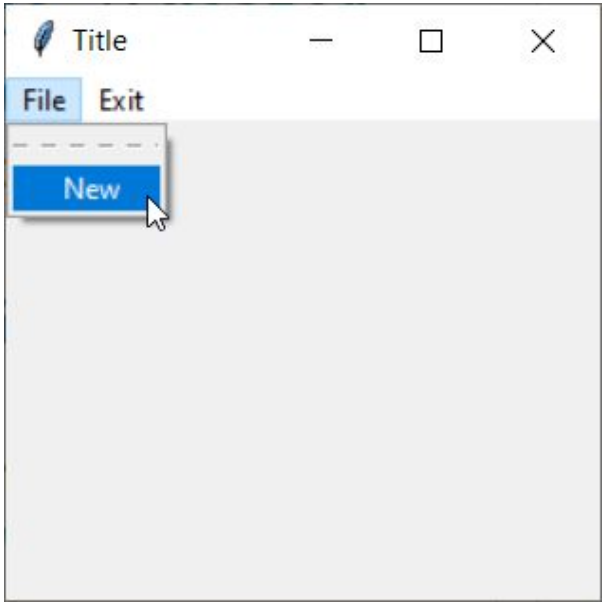
# TkCombobox

```
TkCombobox.new(root) do
  values [1, 2, 3]
  state 'readonly'
  pack(pady: [20,0])
end
```



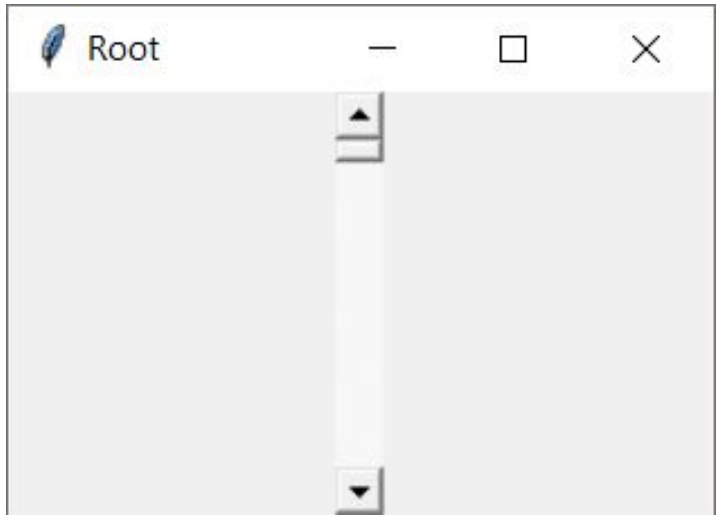
# TkMenu

```
file_menu = TkMenu.new(root)
file_menu.add('command', label: 'New',
                         command: proc { ... },
                         underline: 0)
menu = TkMenu.new(root)
menu.add('cascade', menu: file_menu,
                    label: 'File',
                    underline: 0)
menu.add('command', label: 'Exit',
                    command: proc { ... },
                    underline: 0)
root.menu(menu)
```



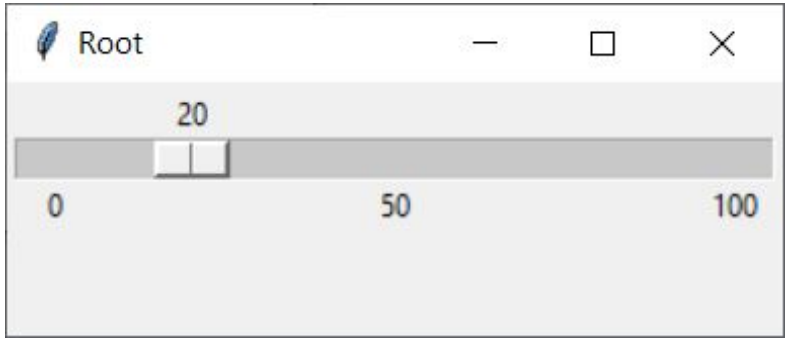
# TkScrollbar

```
TkScrollbar.new do
  command proc { |idx|
    list.yview(*idx)
  }
  pack(side: 'left',
       fill: 'y',
       expand: 1)
end
```



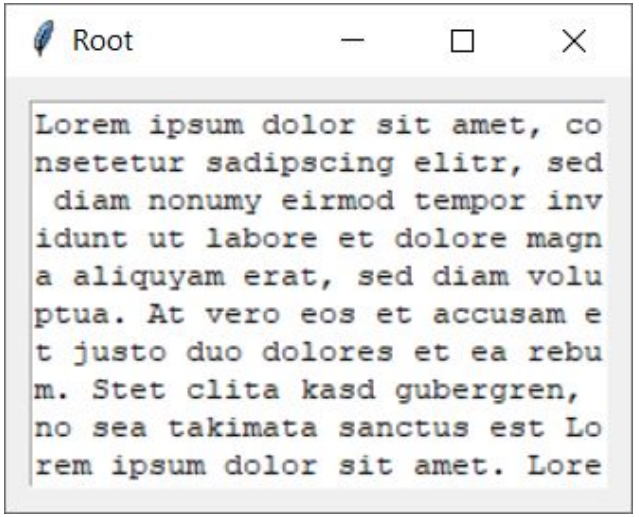
# TkScale

```
TkScale.new do
  orient 'horizontal'
  from 0
  to 100
  length 300
  tickinterval 50
  pack
end
```



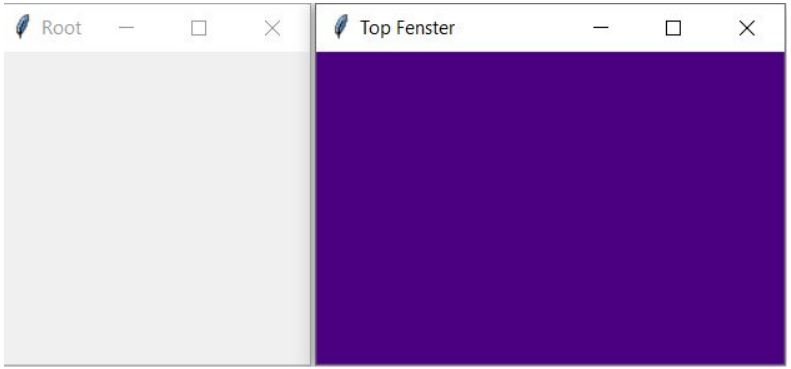
# TkText

```
TkText.new(root) do
  width 30
  height 10
  pack(padx:10, pady:10)
end
```



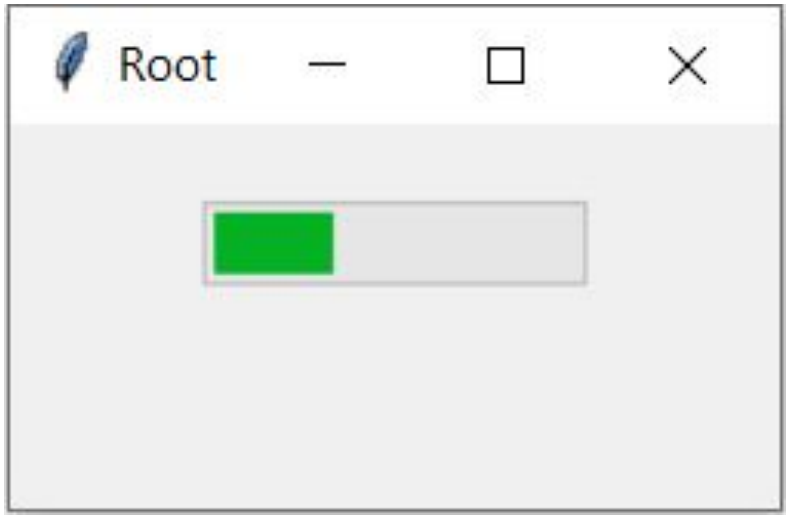
# TkToplevel

```
TkToplevel.new do
  title 'Top Fenster'
  background 'indigo'
  minsize 300, 200
end
```



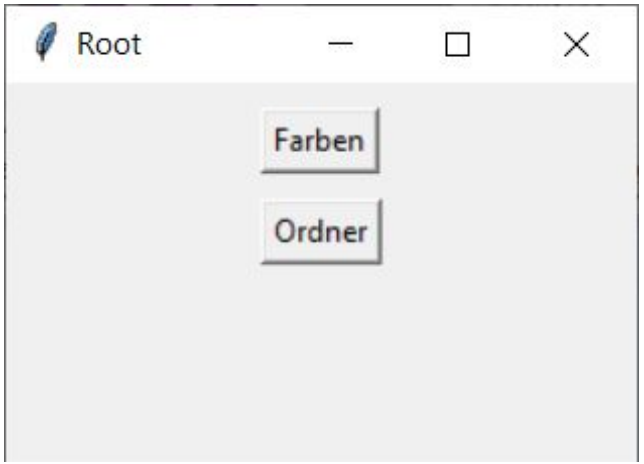
# TkProgressbar

```
var = TkVariable.new(33)
TkProgressbar.new(root) do
  variable var
  maximum 100
  pack(pady: [20,0])
end
```

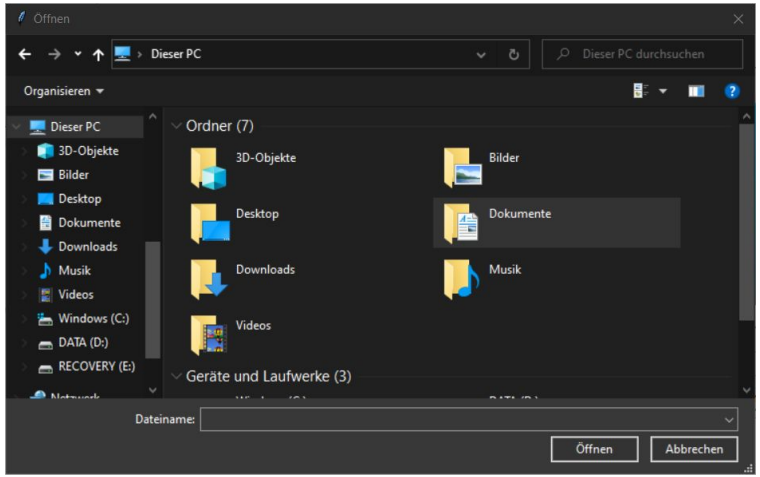


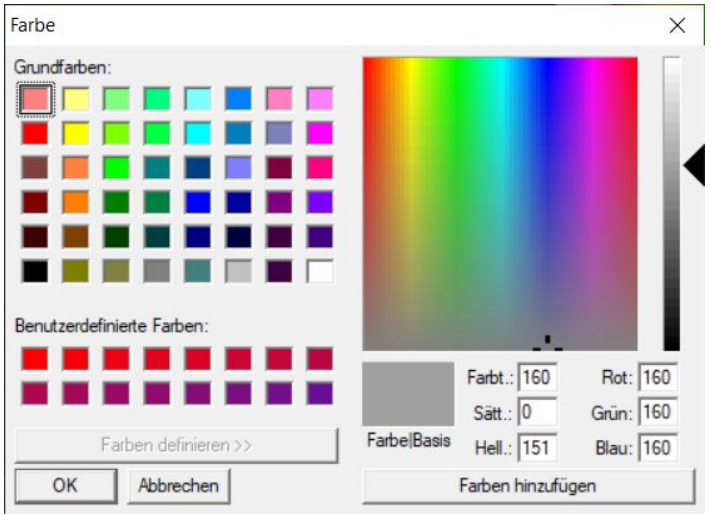
# DialogBox

```
button1_click = proc do
  Tk.chooseColor
end
button2_click = proc do
  Tk.getOpenFile
end

button1 = TkButton.new(root) do
  text 'Farben'
  pack(pady: 10)
end
button2 = TkButton.new(root) do
  text 'Ordner'
  pack(pady: [0,10])
end

button1.command = button1_click
button2.command = button2_click
```



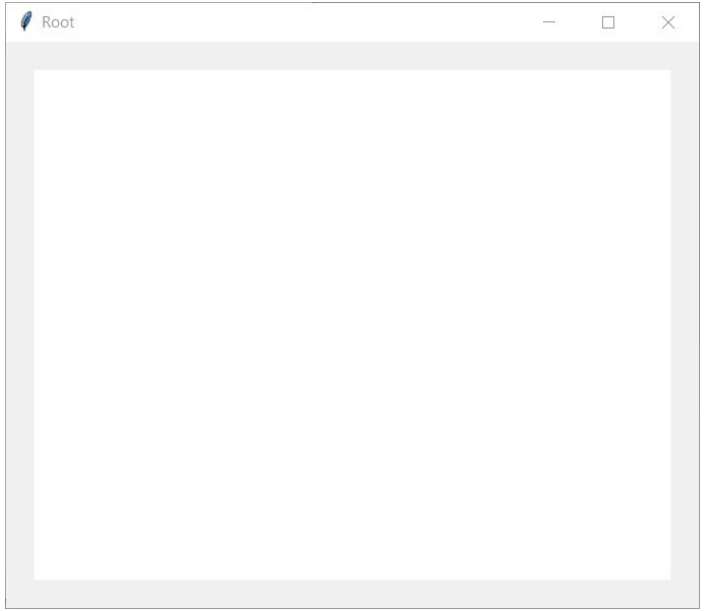
# DialogBox


Öffnen
Dieser PC
Dieser PC durchsuchen
Organisieren
Dieser PC
3D-Objekte
Bilder
Desktop
Dokumente
Downloads
Musik
Videos
Windows (C:)
DATA (D:)
RECOVERY (E:)
Ordner (7)
3D-Objekte
Bilder
Desktop
Dokumente
Downloads
Musik
Videos
Geräte und Laufwerke (3)
Dateiname:
Öffnen
Abbrechen


Farbe
Grundfarben:
Benutzerdefinierte Farben:
Farben definieren >>
OK
Abbrechen
Farbe|Basis
Farbt.: 160
Sätt.: 0
Hell.: 151
Rot: 160
Grün: 160
Blau: 160
Farben hinzufügen

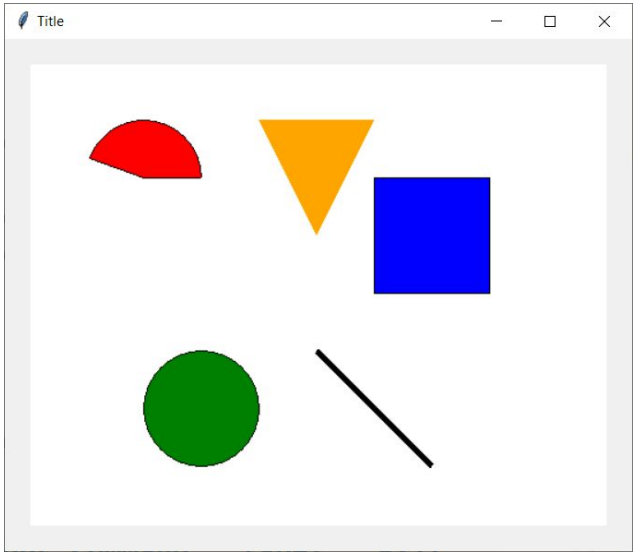
# TkCanvas

```
TkCanvas.new(root) do
  background 'white'
  width 500
  height 400
  pack(padx:20, pady:20)
end
```

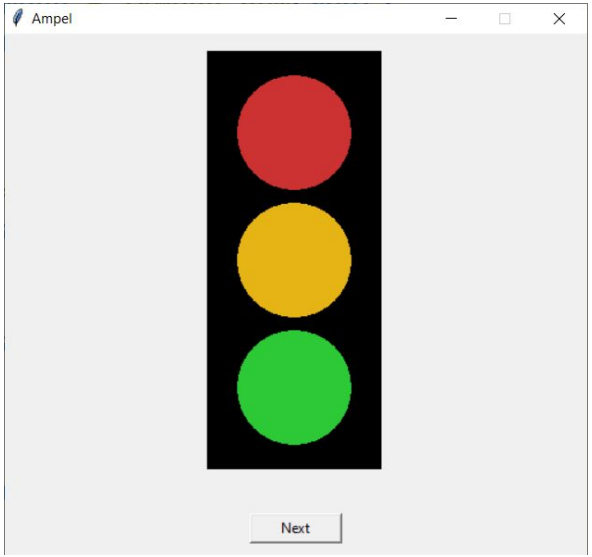


# Canvas Elemente

```
TkcArc.new(canvas, [[50, 50], [150, 150]], start: 0,
                                           extent: 160,
                                           fill: 'red')
TkcLine.new(canvas, [[250, 250], [350, 350]], fill: 'black',
                                              width: 5)
TkcRectangle.new(canvas, [[300, 100], [400, 200]], fill: 'blue')
TkcOval.new(canvas, [[100, 250], [200, 350]], fill: 'green')
TkcPolygon.new(canvas, [[200, 50], [300, 50], [250, 150]], fill: 'orange')
```

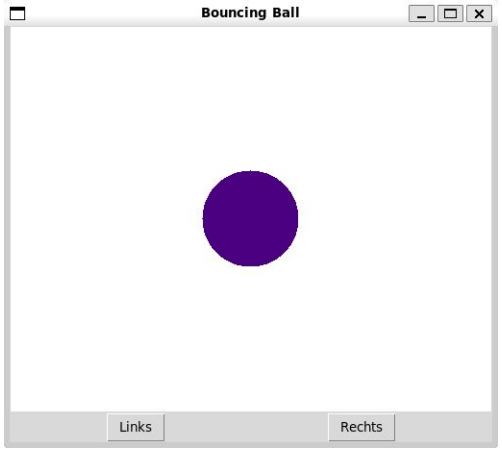


# Ampel



# Übungsaufgaben

# Bouncing Ball




vertikale Bewegung implementieren

**und / oder**

# Christmas Tree

Szene dekorieren und animieren

# Was haben wir heute gelernt

**Was ist grafische Programmierung**

**Tk & andere GUI-Libraries**

**GUIs & kleine Animationen mit Ruby Tk erstellen**

## Tk Dokumentation

https://www.rubydoc.info/stdlib/tk/Tk

## Tk Tutorials/Anleitungen

https://www.tutorialspoint.com/ruby/ruby_tk_guide.htm

https://tkdocs.com/index.html