



# GUI 图形用户界面

车万翔

哈尔滨工业大学



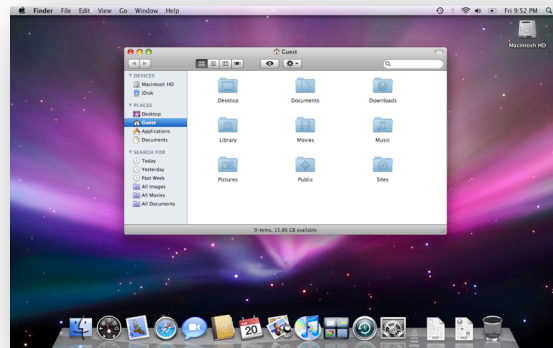
# GUI 的概念



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

- ❖ Graphical User Interface ( GUI ) 图形用户界面或图形用户接口
- ❖ 图形方式显示的计算机操作环境，与命令行界面相比更为简便易用
- ❖ 极大地方便了非专业用户使用，不再需要死记硬背大量的命令，取而代之的是通过窗口、菜单、按钮等方式来方便地进行操作

```
jim — bash — 103x20
Last login: Tue Sep 25 13:00:56 on ttys000
jim-Hoskins-Mac:~ jim$ ls -la
total 8
drwxr-xr-x+ 13 jim  staff   442 Sep 25 13:00 .
drwxr-xr-x+  6 root   admin  204 Sep 25 12:58 ..
-rw-r-----  1 jim  staff    3 Sep 25 12:58 .CPUTextEncoding
drwx-----  2 jim  staff   60 Sep 25 12:52 .Trash
-rw-r-----  1 jim  staff   57 Sep 25 13:00 .bash_history
drwx-----+ 10 jim  staff  348 Sep 25 12:57 Desktop
drwx-----+  4 jim  staff  136 Sep 25 12:58 Documents
drwx-----+  4 jim  staff  136 Sep 25 12:58 Downloads
drwx-----+ 83 jim  staff 1122 Sep 25 12:58 Library
drwx-----+  3 jim  staff  182 Sep 25 12:58 Movies
drwx-----+  3 jim  staff  182 Sep 25 12:58 Music
drwx-----+  4 jim  staff  136 Sep 25 12:58 Pictures
drwxr-xr-x+  5 jim  staff  178 Sep 25 12:58 Public
jim-Hoskins-Mac:~ jim$ Where am I?
```





# GUI 发展历史



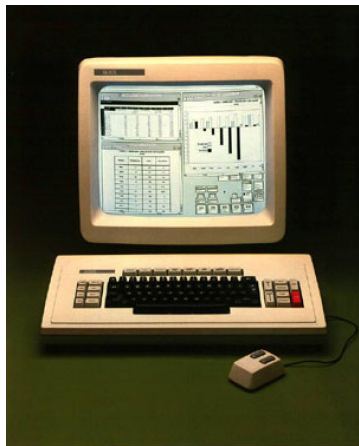
哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY



1962发明了世界上第一个鼠标



1984年，苹果推出了Macintosh



1981年，施乐公司制造出基于GUI的计算机



1985年，微软推出了Windows 1.0



## ❖ 如何编写GUI程序？

## ❖ 第三方 API 库

- Tkinter、wxPython、PyQt、PyGtk

## ❖ Tkinter (Tk Interface)

- Python 的标准 Tk GUI 工具包接口

## ❖ 搭积木

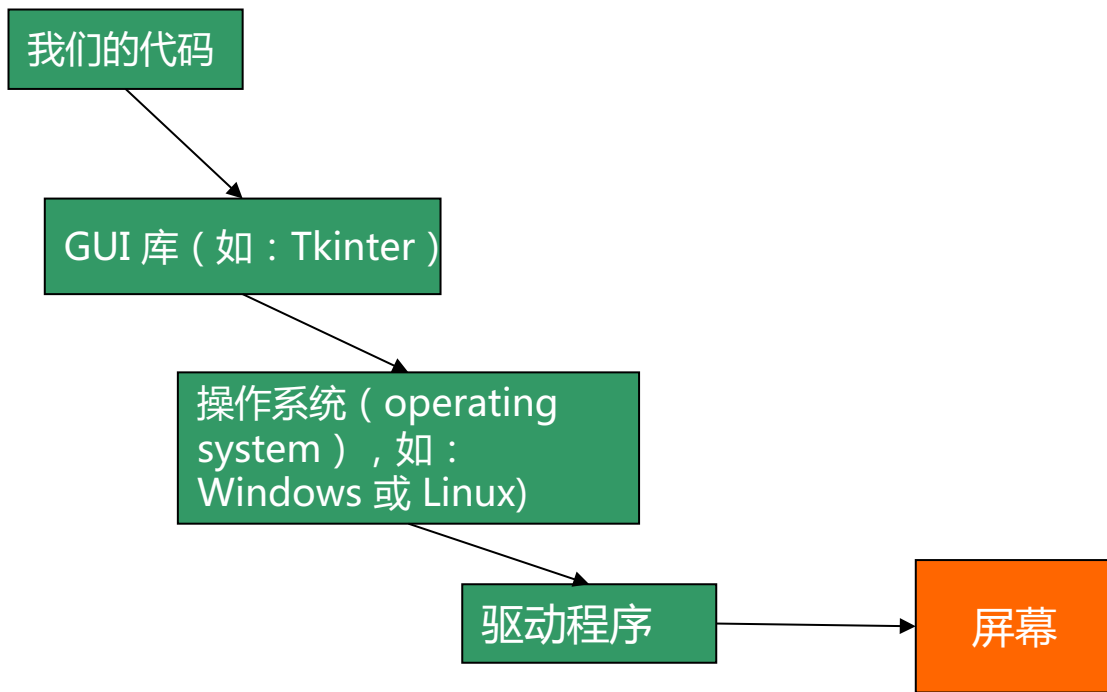




# 软件抽象层级



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY





# GUI 常用组件 (Widget)



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

## ❖ Label

- 不可编辑的单行文本



## ❖ Button

- 按钮



## ❖ Checkbutton

- 选择按钮



## ❖ Entry

- 文本输入框





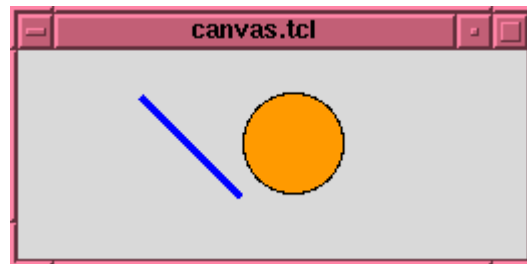
# GUI 常用组件 (Widget)



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

## ❖ Canvas

- 画布



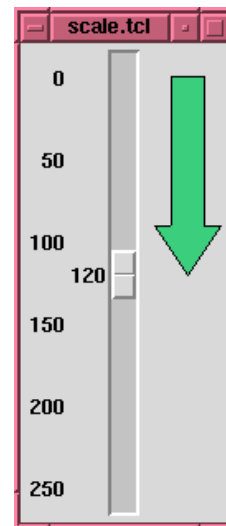
## ❖ Listbox

- 列表框



## ❖ Scale

- 滑动条





# Hello, Tkinter

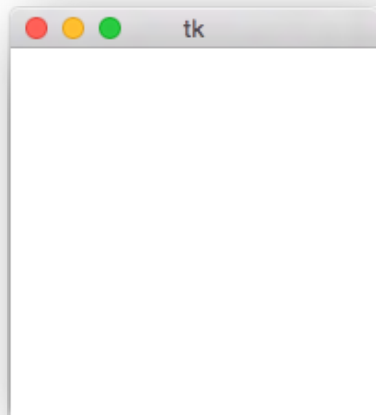


哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

```
import tkinter # Import tkinter
```

```
root = tkinter.Tk() # Create a root window
```

```
root.mainloop() # Create an event loop
```







# GUI 组件添加的方法



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

- ❖ 导入模块
- ❖ 创建主窗口
- ❖ 创建组件对象，由Tkinter的该组件相关方法来实现的（关于组件的具体使用可以 Google 一下）
- ❖ 用 pack（）（布局器）来管理和显示组件
- ❖ 主窗口调用mainloop（）进入主事件循环



# 简单组件示例



```
import tkinter # Import tkinter
```

```
root = tkinter.Tk() # Create a root window
```

```
# Create a label
```

```
label = tkinter.Label(root, text = "Welcome to Python")
```

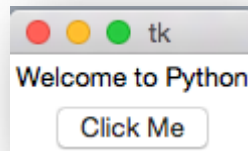
```
# Create a button
```

```
button = tkinter.Button(root, text = "Click Me")
```

```
label.pack() # Display the label in the window
```

```
button.pack() # Display the button in the window
```

```
root.mainloop() # Create an event loop
```





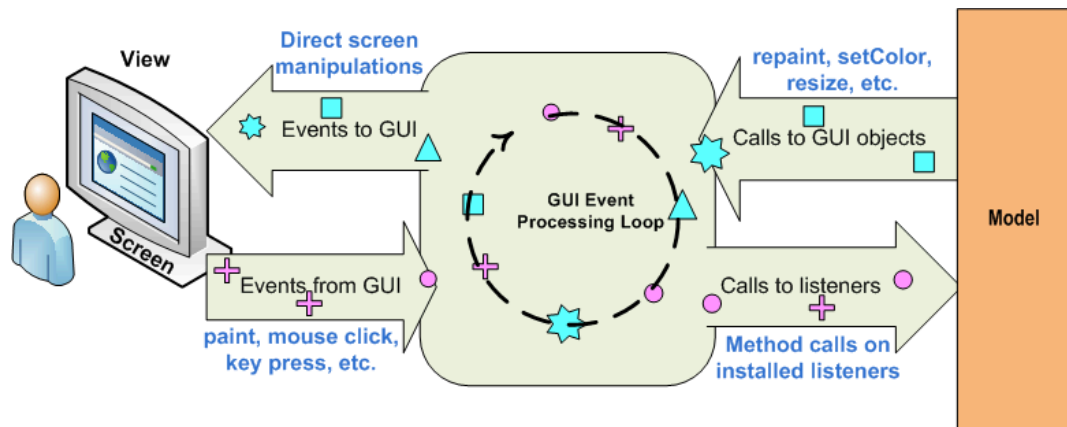
# 事件 ( Events ) 处理



`root.mainloop()` # Create an event loop

## ❖ 生成一个事件处理循环，持续处理事件

- 事件是当受控对象状态改变时，给应用程序的通知或消息
- 如按下按钮、鼠标的移动、按键盘等





# 按钮事件的处理



```
from tkinter import * # Import tkinter
```

```
def processOK():  
    print("OK button is clicked")
```

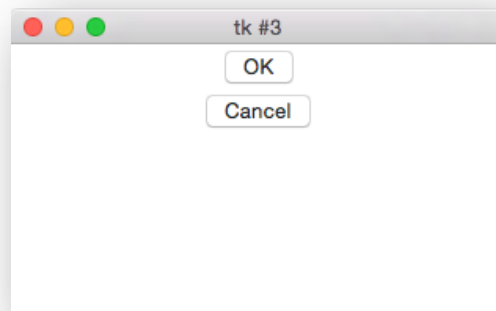
```
def processCancel():  
    print("Cancel button is clicked")
```

```
root = Tk() # Create a root window
```

```
btOK = Button(root, text = "OK", command = processOK)  
btCancel = Button(root, text = "Cancel", command = processCancel)
```

```
btOK.pack() # Place the button in the window  
btCancel.pack() # Place the button in the window
```

```
root.mainloop() # Create an event loop
```



回调函数  
callback



`widget.bind(event, handler)`

- ❖ 事件处理 ( Event handlers ) 是捕获并响应事件的用户函数
- ❖ 绑定或捆绑 ( bind ) 建立事件与相应事件处理函数联系



# Events



Event	Description
<code>&lt;Button-<i>i</i>&gt;</code>	Button-1, Button-2, and Button-3 are for left, middle, or right buttons. When a mouse button is pressed over the widget, Tkinter automatically grabs the mouse pointer location. ButtonPressed- <i>i</i> is synonymous to Button- <i>i</i> .
<code>&lt;Bi-Motion&gt;</code>	An event occurs, when a mouse button is moved while being held down on the widget.
<code>&lt;ButtonReleased-<i>i</i>&gt;</code>	An event occurs, when a mouse button is released.
<code>&lt;Double-Button-<i>i</i>&gt;</code>	An event occurs, when a mouse button is double-clicked.
<code>&lt;Triple-Button-<i>i</i>&gt;</code>	An event occurs, when a mouse button is triple-clicked.
<code>&lt;Enter&gt;</code>	An event occurs, when a mouse pointer enters the widget.
<code>&lt;Leave&gt;</code>	An event occurs, when a mouse pointer leaves the widget.
<code>&lt;Return&gt;</code>	An event occurs, when the Enter key is pressed. You can bind any key such as <code>&lt;A&gt;</code> , <code>&lt;B&gt;</code> , <code>&lt;Up&gt;</code> , <code>&lt;Down&gt;</code> , <code>&lt;Left&gt;</code> , <code>&lt;Right&gt;</code> in the keyboard with an event.
<code>&lt;Key&gt;</code>	An event occurs, when a key is pressed.
<code>&lt;Shift-A&gt;</code>	An event occurs, when the Shift+A keys are pressed. You use Alt, Shift, and Control to combine with other keys.



# Event 属性



Event	Description
<code>widget</code>	The widget object that fires this event.
<code>x</code> and <code>y</code>	The current mouse location in the widget pixels.
<code>x__root</code> and <code>y__root</code>	The current mouse position relative to the upper left corner of the screen, in pixels.
<code>num</code>	The button number (1, 2, 3), indicating which mouse button was clicked.
<code>char</code>	The char entered from the keyboard for key events.
<code>keysym</code>	The key symbol for the key entered from the keyboard for key events.
<code>keycode</code>	The key code for the key entered from the keyboard for key events.



# 鼠标、键盘事件 Demo



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

```
from tkinter import *
```

```
root = Tk()
```

```
def key(event):  
    print("pressed", event.char)
```

```
def button(event):  
    frame.focus_set()  
    print("clicked at", event.x, event.y)
```

```
frame = Frame(root, width=100, height=100)  
frame.bind("<Key>", key)  
frame.bind("<Button-1>", button)  
frame.pack()
```

```
root.mainloop()
```



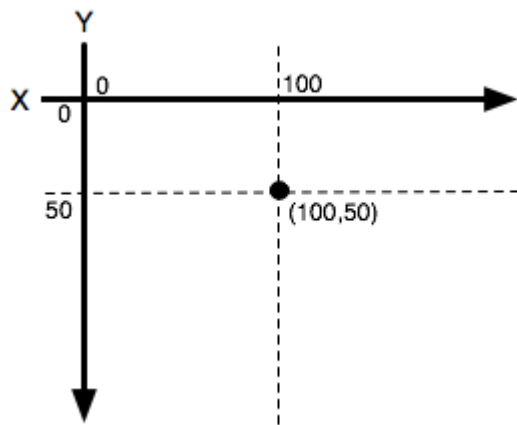


# 画布 (Canvas)



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

- ❖ 显示点、线、矩形、圆等各种形状
- ❖ 像素的坐标系





# Canvas Demo



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

# Display a rectangle

```
def displayRect():  
    canvas.create_rectangle(10, 10, 190, 90, tags = "rect")
```

# Display an oval

```
def displayOval():  
    canvas.create_oval(10, 10, 190, 90, fill = "red", tags = "oval")
```

# Display a line

```
def displayLine():  
    canvas.create_line(10, 10, 190, 90, fill = "red", tags = "line")
```

# Display a string

```
def displayString():  
    canvas.create_text(60, 40, text = "Hi, I am a string", font = "bold underline", tags = "string")
```

# Clear drawings

```
def clearCanvas():  
    canvas.delete("rect", "oval", "arc", "line", "string")
```



# Canvas Demo

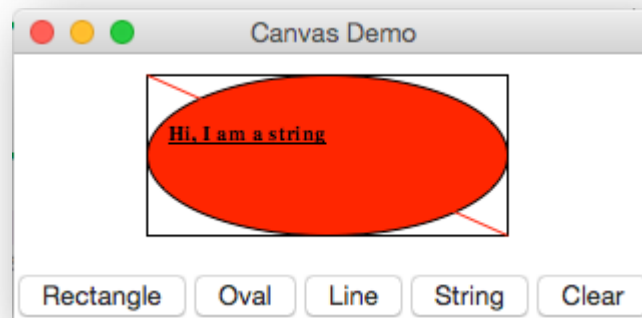


哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

```
window = Tk()
window.title("Canvas Demo") # Set title
# Place canvas in the window
canvas = Canvas(window, width = 200, height = 100, bg = "white")
canvas.pack()
# Place buttons in frame
frame = Frame(window)
frame.pack()
btRectangle = Button(frame, text = "Rectangle", command = displayRect)
btOval = Button(frame, text = "Oval", command = displayOval)
btLine = Button(frame, text = "Line", command = displayLine)
btString = Button(frame, text = "String", command = displayString)
btClear = Button(frame, text = "Clear", command = clearCanvas)
```

```
btRectangle.grid(row = 1, column = 1)
btOval.grid(row = 1, column = 2)
btLine.grid(row = 1, column = 3)
btString.grid(row = 1, column = 4)
btClear.grid(row = 1, column = 5)
```

```
window.mainloop()
```

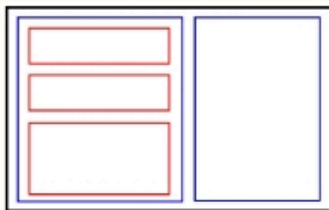




## ❖ 三种方式

- Pack
- Grid
- Place

Pack



Place



Grid

0, 0		
1, 0	1, 1	1, 2
2, 0		2, 2
3, 0		3, 2
4, 0		



# 综合应用：动画



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

```
from tkinter import * # Import tkinter
```

```
def increaseCircle(event):
```

```
    pass
```

```
def decreaseCircle(event):
```

```
    pass
```

```
window = Tk()
```

```
window.title("Control Circle Demo") # Set a title
```

```
canvas = Canvas(window, bg = "white", width = 200, height = 200)
```

```
radius = 50
```

```
canvas.create_oval(100 - radius, 100 - radius, 100 + radius, 100 + radius, tags = "oval")
```

```
canvas.bind("<Up>", increaseCircle)
```

```
canvas.bind("<Down>", decreaseCircle)
```

```
canvas.focus_set()
```

```
canvas.pack()
```

```
window.mainloop()
```



```
def increaseCircle(event):
```

```
    canvas.delete("oval")
```

```
    global radius
```

```
    if radius < 100:
```

```
        radius += 2
```

```
    canvas.create_oval(100 - radius, 100 - radius, 100 + radius, 100 + radius, tags = "oval")
```

```
def decreaseCircle(event):
```

```
    canvas.delete("oval")
```

```
    global radius
```

```
    if radius > 2:
```

```
        radius -= 2
```

```
    canvas.create_oval(100 - radius, 100 - radius, 100 + radius, 100 + radius, tags = "oval")
```



# 随机点名



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

```
import tkinter as tk
import random

def run():
    global start
    global job

    if not start:
        start = True
        button.configure(text = 'Stop')
        on_timer()
    else:
        start = False
        button.configure(text = 'Start')
        root.after_cancel(job)

def on_timer():
    global job
    student = random.choice(students)
    label.configure(text = student)
    job = root.after(100, on_timer)

root = tk.Tk()
root.title('Random Selection')
root.geometry('400x200')

start = False

button = tk.Button(text = 'Start', font = "Verdana 26 bold", command = run)
button.pack(side = 'bottom')

label = tk.Label(text = '', font = "Verdana 26 bold")
label.pack()

students = open('list.txt').readlines()

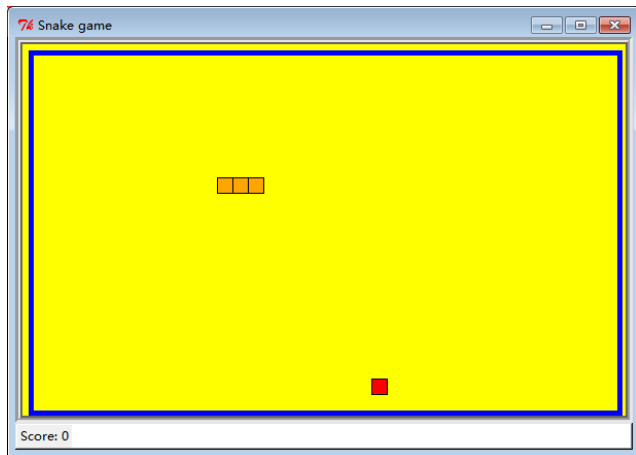
root.mainloop()
```



# 贪吃蛇游戏

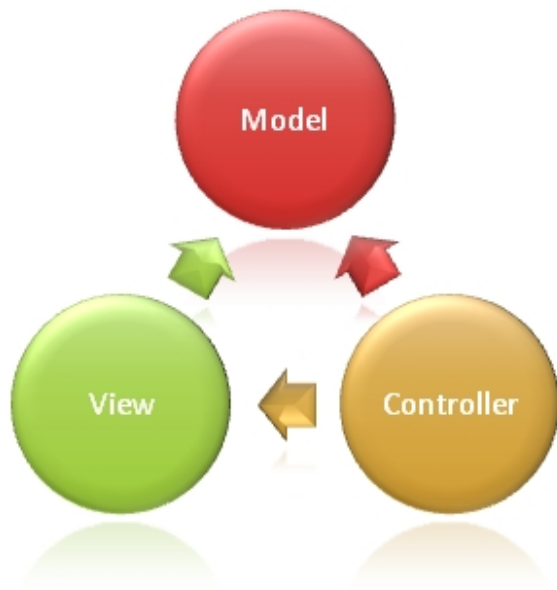


哈尔滨工业大学  
HARBIN INSTITUTE OF TECHNOLOGY



- ❖ 游戏者按任意键进入游戏。游戏者用←、↓、→、↑键来控制蛇在游戏场景内运动，每吃到一个食物，游戏者得10分，分数累加结果会在计分板上显示；与此同时蛇身长出一节。当贪吃蛇的头部撞击到游戏场景边框或者蛇的身体时游戏结束，并显示游戏者最后得分





❖ MVC是一个框架模式，将应用程序被分成三个核心部件：

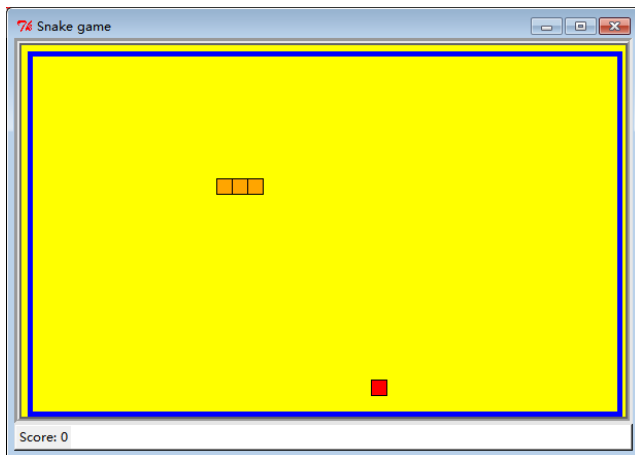
- 模型 ( Model )
- 视图 ( View )
- 控制器 ( Control )



# 贪吃蛇游戏的MVC设计



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY



## Model

- random\_food()
- snake()
- score()

## View

- draw\_wall()
- draw\_food()
- draw\_snake()
- show\_score()

## Control

- play()
- iseated()
- isdead()
- move()
- gameover()
- restart()