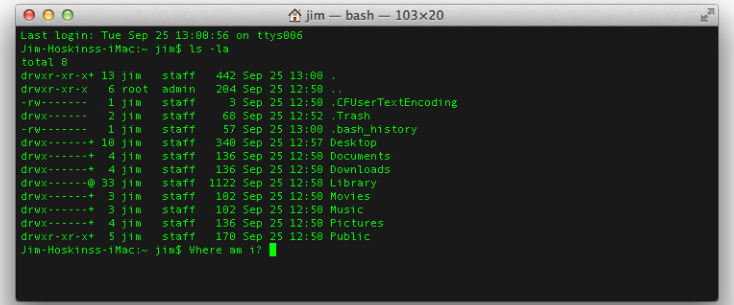
GUI 图形用户界面

车万翔

哈尔滨工业大学

GUI 的概念

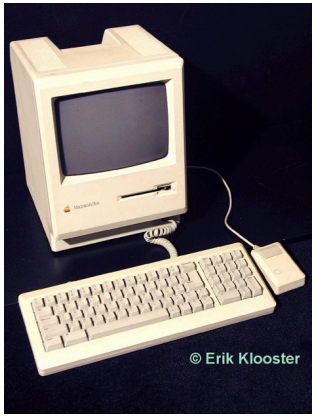
- Graphical User Interface（GUI）图形用户界面或图形用户接口
- 图形方式显示的计算机操作环境，与命令行界面相比更为简便易用
- 极大地方便了非专业用户使用，不再需要死记硬背大量的命令，取而代之的是通过窗口、菜单、按钮等方式来方便地进行操作



GUI 发展历史

1962发明了世界上第一个鼠标

1981年，施乐公司制造出基于GUI的计算机



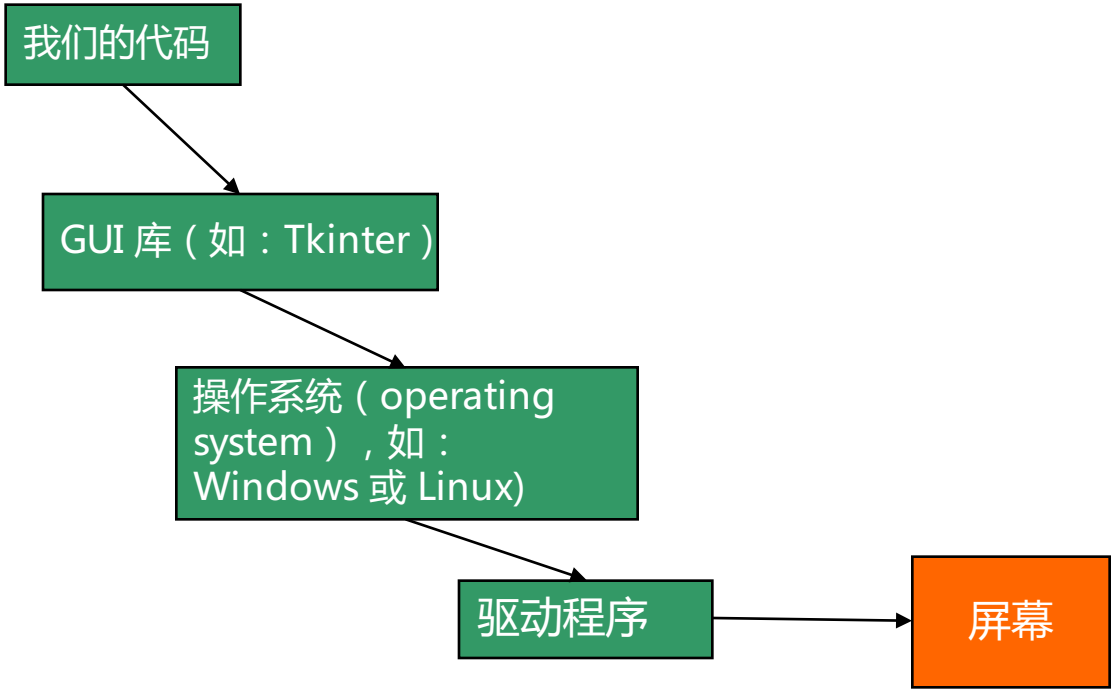
1984年，苹果推出了Macintosh

1985年，微软推出了Windows 1.0

GUI 编程基础

- **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


我们的代码
GUI 库（如：Tkinter）
操作系统（operating system），如：Windows 或 Linux)
驱动程序
屏幕

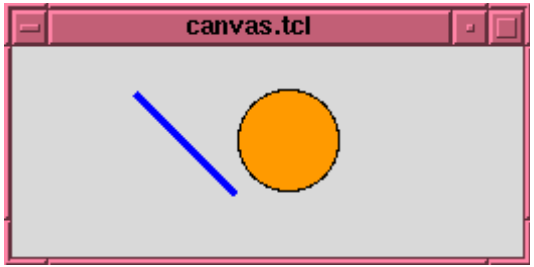
GUI 常用组件 (Widget)

- **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)

Canvas

- 画布

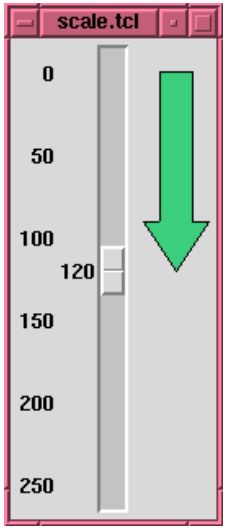


Listbox

- 列表框

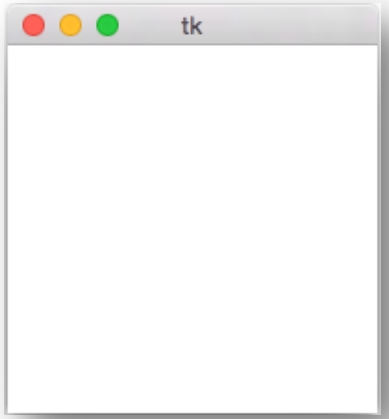
Scale

- 滑动条



Hello, Tkinter


```
import Tkinter # Import tkinter

root = Tkinter.Tk() # Create a root window

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

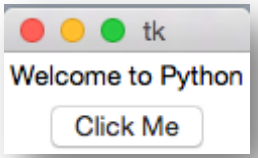


GUI 组件添加的方法

- 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

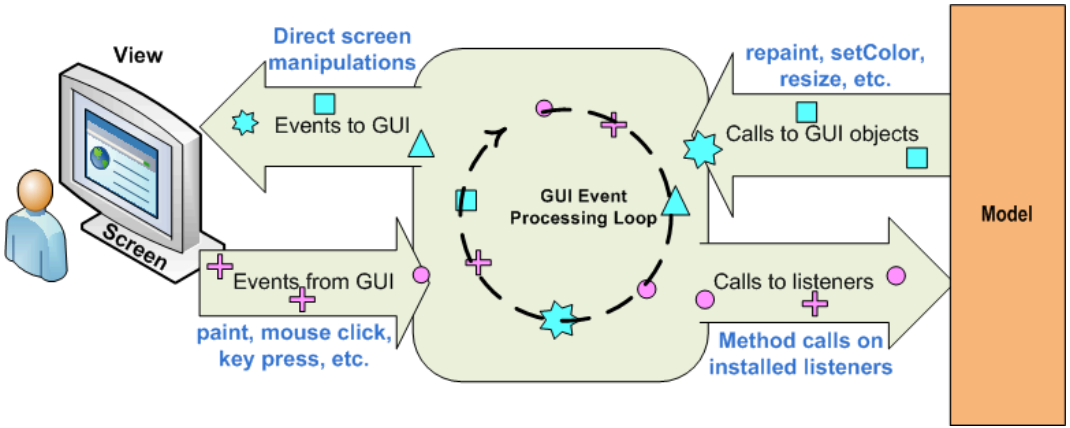

```
import Tkinter # Import tkinter

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

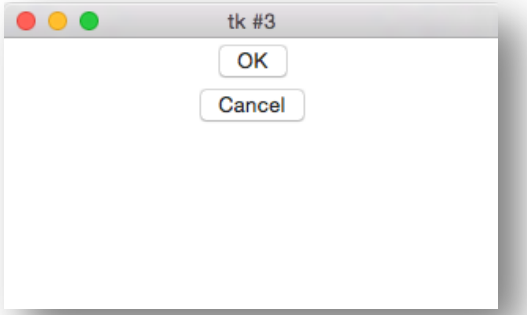
root.mainloop() # Create an event loop

- **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
<Button-*i*>	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 is synonymous to Button-i.
<B*i*-Motion>	An event occurs, when a mouse button is moved while being held down on the widget.
<ButtonReleased-*i*>	An event occurs, when a mouse button is released.
<Double-Button-*i*>	An event occurs, when a mouse button is double-clicked.
<Triple-Button-*i*>	An event occurs, when a mouse button is triple-clicked.
<Enter>	An event occurs, when a mouse pointer enters the widget.
<Leave>	An event occurs, when a mouse pointer leaves the widget.
<Return>	An event occurs, when the Enter key is pressed. You can bind any key such as <A>, <B>, <Up>, <Down>, <Left>, <Right> in the keyboard with an event.
<Key>	An event occurs, when a key is pressed.
<Shift-A>	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
widget	The widget object that fires this event.
x and y	The current mouse location in the widget pixels.
x__root and y_root	The current mouse position relative to the upper left corner of the screen, in pixels.
num	The button number (1, 2, 3), indicating which mouse button was clicked.
char	The char entered from the keyboard for key events.
keysym	The key symbol for the key entered from the keyboard for key events.
keycode	The key code for the key entered from the keyboard for key events.

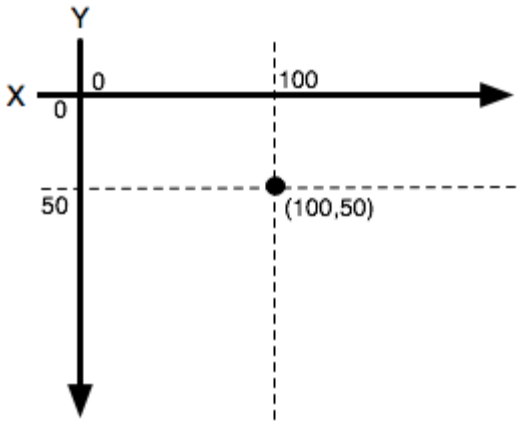
鼠标、键盘事件 Demo

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）

- 显示点、线、矩形、圆等各种形状
- 像素的坐标系



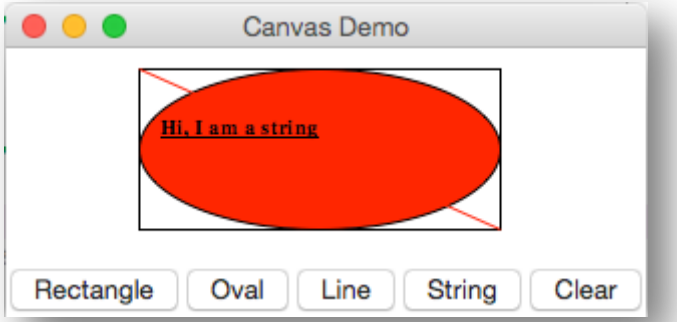
Canvas Demo

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

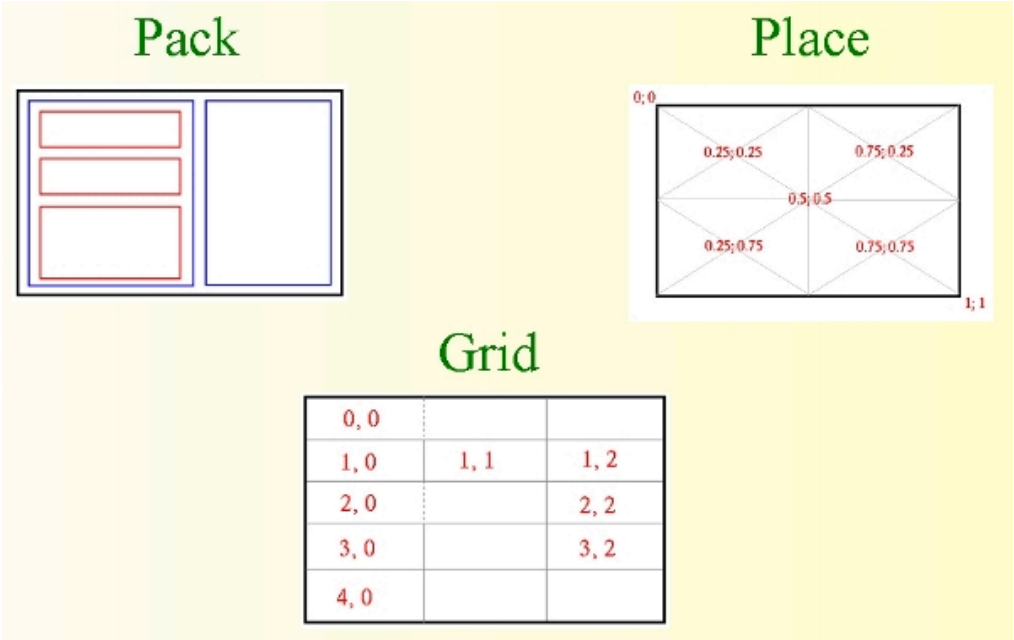
```
window = Tk()
window.title("Canvas Demo") # Set title
# Place canvas in the window
canvas = Canvas(window, width = 200, height = 100, bg = "white")
canvas.pack()
# Place buttons in frame
frame = Frame(window)
frame.pack()
btRectangle = Button(frame, text = "Rectangle", command = displayRect)
btOval = 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()
```



组件位置管理

三种方式

- Pack
- Grid
- Place



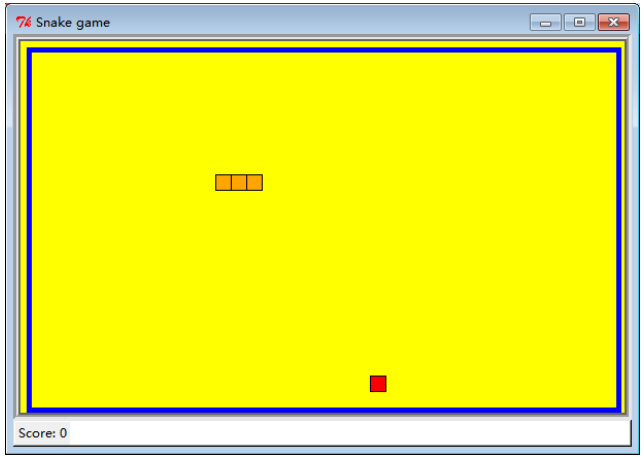
综合应用：动画


```
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()
```

综合应用：动画

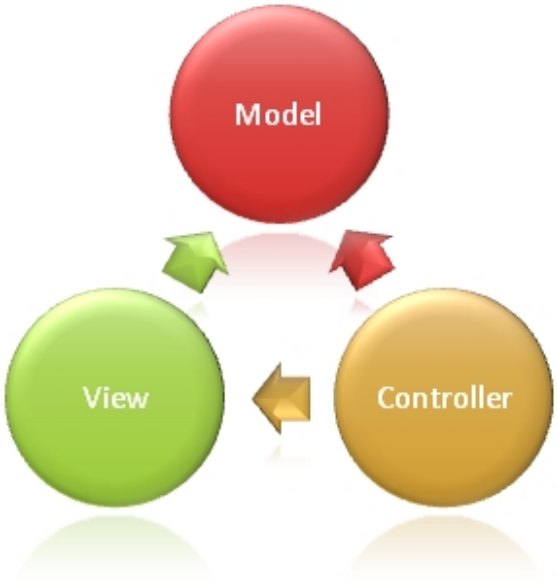

```
def increaseCircle(event):
    canvas.delete("oval")
    global radius
    if radius < 100:
        radius += 2
    canvas.create_oval(100 - radius, 100 - radius, 100 + radius, 100 + radius, tags = "oval")

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")
```

贪吃蛇游戏



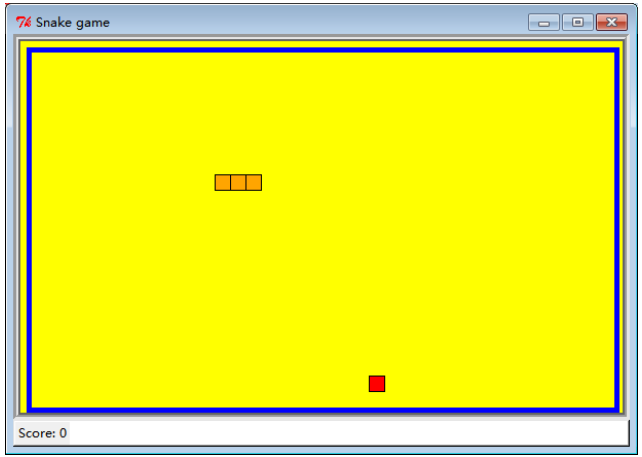
- 游戏者按任意键进入游戏。游戏者用←、↓、→、↑键来控制蛇在游戏场景内运动，每吃到一个食物，游戏者得10分，分数累加结果会在计分板上显示；与此同时蛇身长出一节。当贪吃蛇的头部撞击到游戏场景边框或者蛇的身体时游戏结束，并显示游戏者最后得分

MVC模式



- 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



View

- draw_wall()
- draw_food()
- draw_snake()
- show_score()

Model

- random_food()
- snake()
- score()

具体代码参见 **snake.py** 需要自学类、对象等内容

Control

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