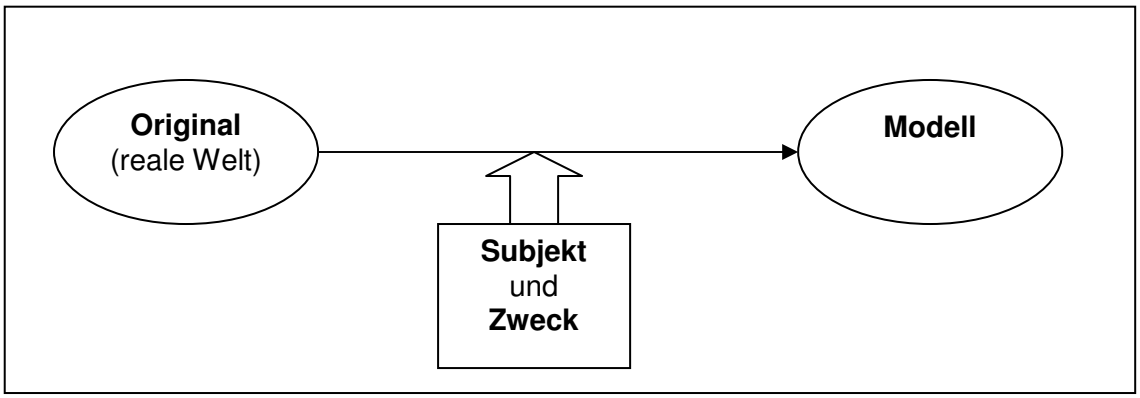
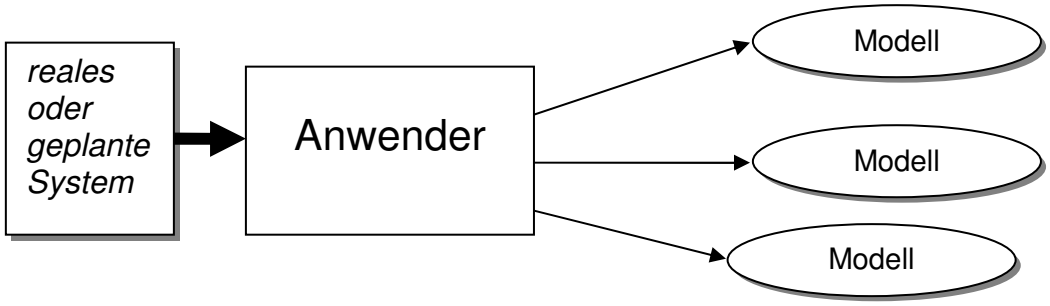
Ergebnisse der Seminarsitzung vom 14. Januar 2008

Definition 1
Ein Modell ist eine struktur- und verhaltenstreue Beschreibung eines existierenden oder geplanten Systems.

Definition 2
Ein Modell ist eine vereinfachte Nachbildung eines geplanten oder real existierenden Originalsystems mit seinen Prozessen in einem anderen begrifflichen oder gegenständlichen System. Es unterscheidet sich hinsichtlich der untersuchungsrelevanten Eigenschaften nur innerhalb eines vom Untersuchungsziel abhängigen Toleranzrahmens vom Vorbild.



Wesentliche Eigenschaft eines Modells:
Je nach Zweck und Subjekt kann es zu einem realen System mehrere Realisierungen geben.



Entwicklung des algorithmischen Kerns eines komplexen Systems mithilfe der Zustandsmodellierung am Beispiel des Spiels ***Senso***. Die Entwicklung des Spiels teilt sich in zwei Teile. Die

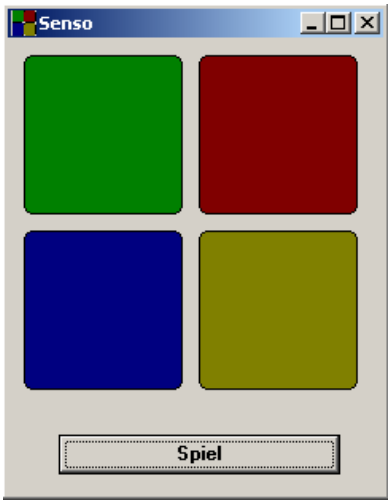
- **Grafik** wird ***objektorientiert***, der
- **Spielkern *zustandsorientiert***

entwickelt.

Teil 1: Grafik

Aufgabe 1
Starte das Spiel „Senso“ und teste es.

Aufgabe 2
Die vier farbigen Tasten haben einige Gemeinsamkeiten. Schreibe die Gemeinsamkeiten für jede Taste in die vorgegebene Box.



gelbe Taste	rote Taste
grüne Taste	blaue Taste

Aufgabe 3

a) Die obige Darstellung nennt sich **Klassendiagramm**. Beschreibe, welche Entwurfselemente sich im Klassendiagramm wiederfinden.

b) Was könnte der Pfeil und die Zahlen am Pfeil bedeuten?

Aufgabe 4
Sieh dir die Implementierung des Klassendiagramms an.

```
type
  TTaste = class
     x, y : integer;
     height, width : integer;
     Farbedunkel : TColor;
     Farbehell: TColor;
  end;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Timer1: TTimer;
    […]
  private
    { Private-Deklarationen }
  public
    { Public-Deklarationen }
    gruen, gelb, rot, blau : TTaste;
    procedure zeichneHell (o: TTaste);
    procedure zeichneDunkel (o: TTaste);
  end;

var
  Form1: TForm1;

implementation
{$R *.dfm}

procedure TForm1.zeichneHell (o: TTaste);
begin
  //
end;

procedure TForm1.zeichneDunkel (o: TTaste);
begin
   //
end;
```

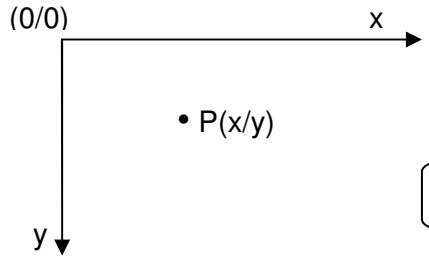
a) Beschreibe die Umsetzung des Diagramms in Delphi-Code.

b) Was könnte (o: TTaste) in den Zeichne-Prozeduren bedeuten?

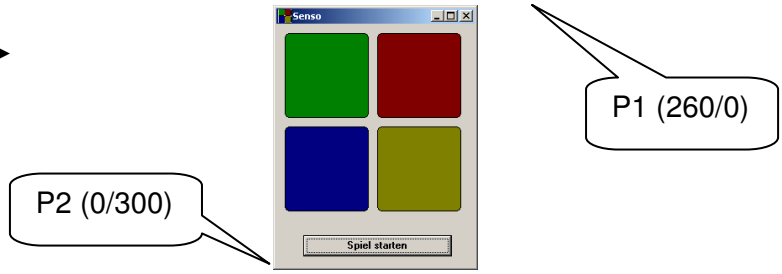
c) Warum gibt es zwei Zeichne-Prozeduren?

Aufgabe 5
Beim Zeichnen wird in Delphi folgendes Koordinatensystem verwendet. Der Koordinatenursprung liegt „links oben“, der positive x-Bereich geht nach rechts, der positive y-Bereich nach unten ab.

Skizze Koordinatensystem

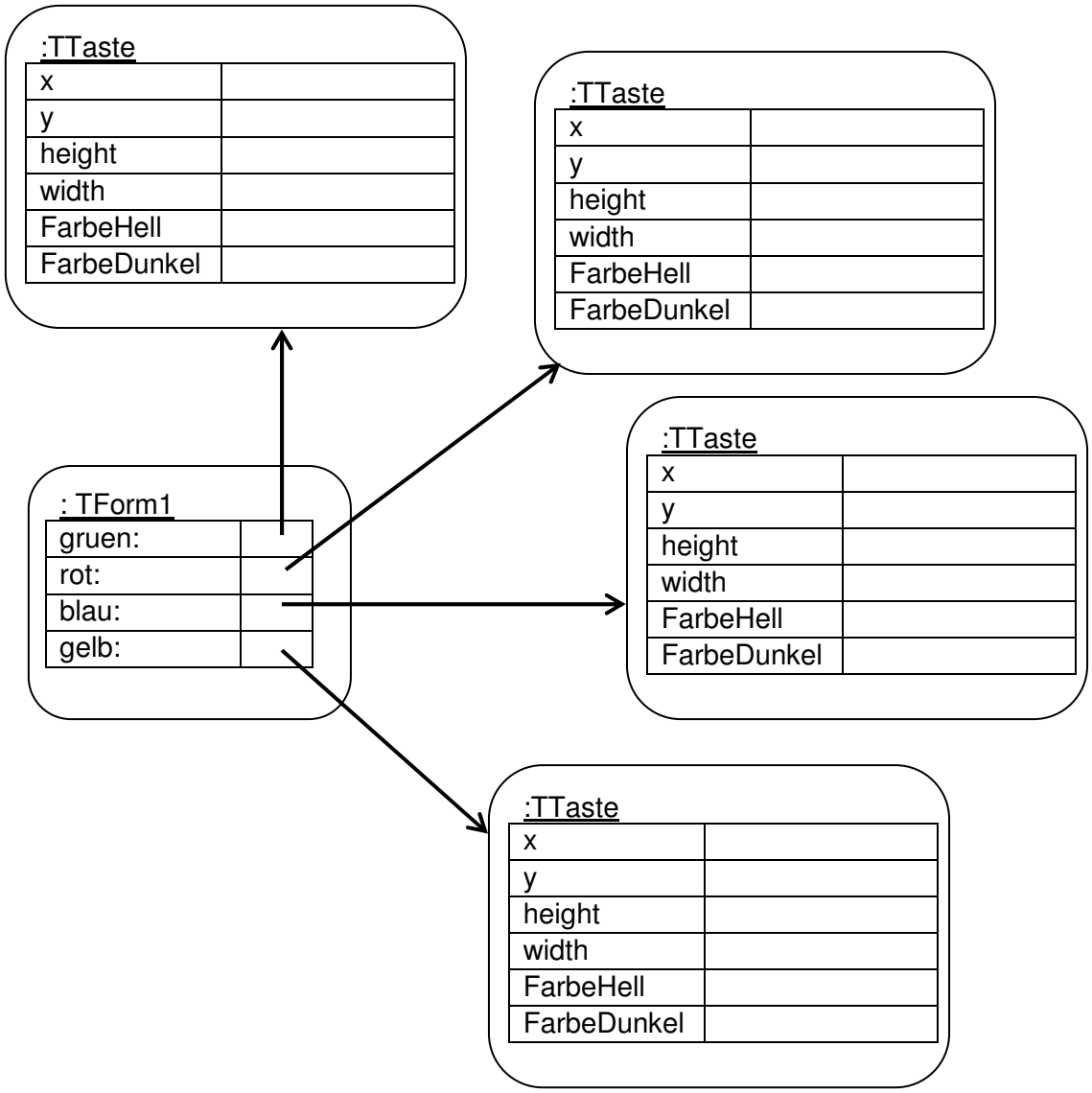


Formulardaten



a) Was zeigt das unten abgebildete Diagramm (**Objektdiagramm**)?

b) Ergänze die Belegungen der Variablen (**Attribute**) der einzelnen Schaltflächen im Objektdiagramm:



Aufgabe 6
Bevor die Tasten tatsächlich existieren, müssen sie

- in der *Form.Create-Methode* erzeugt und
- ihre Attribute mit Daten belegt

werden.

```
procedure TForm1.FormCreate(Sender: TObject);
var i : integer;
begin
   gruen := TTaste.create(10,10,100,100,clgreen,cllime);
   rot := TTaste.create(120,10,100,100,clmaroon,clred);
   blau := TTaste.create(10,120,100,100,clnavy,clblue);
   gelb := TTaste.create(120,120,100,100,clolive,clyellow);
end;
```

a) Ergänze die *Form.Create-Methode*.

b) Nun wird der Klassendeklaration *TTaste* ein **Constructor** hinzugefügt. Ein **Constructor** entspricht einer Prozedur (Methode), die beim ERstellen eines Objekt automatisch ausgeführt wird.

```
type
  TTaste = class
     x, y : integer;
     […]
     constructor create (x, y, height, width : integer; FarbeD, FarbeH :
TColor);
  end;

[…]

implementation
{$R *.dfm}

constructor TTaste.create (x, y, height, width : integer; FarbeD, FarbeH :
TColor);
begin
  self.x := x;




end;
```

Ergänze den Code.

Aufgabe 7
a) Viele Delphi-Komponenten haben eine Zeichenfläche (Canvas) implementiert. Schau in der Delphi-Hilfe die Begriffe : `Canvas, Brush, Rectangle, RoundRec` nach.

b) Die Zeichen-Prozedur für eine dunkle Schaltfläche sieht so aus:

```
procedure TForm1.zeichneDunkel (o: TTaste);
begin
   Form1.canvas.Brush.Color := o.FarbeDunkel;
   Form1.Canvas.RoundRect(o.x, o.y, o.x+o.width, o.y+o.height,10,10);
end;
```

Ergänze die Prozedur für eine helle Schaltfläche!
c) Die Grafikimplementierung kann nun vervollständigt werden.
Implementiere einen *Button*, der die vier Zeichenflächen dunkel einzeichnet.

Teil 2: Der algorithmische Kern

Aufgabe 1
Zunächst sollen die Farben, die nacheinander aufleuchten, in einem Array mit 100 Feldern (zufällig erzeugt) gespeichert werden.

[i]	1	2	3	4	5	…	100
TColor	clmaroon	clnavy	clmaroon	clolive	clgreen	…	clolive

a) Das Array wird als Attribut in der Klasse **TForm1** deklariert:

```
type
  TForm1 = class(TForm)
    […]
  public
    gruen, gelb, rot, blau : TTaste;
    a : array [1..100] of TColor;
    […]
  end;
```

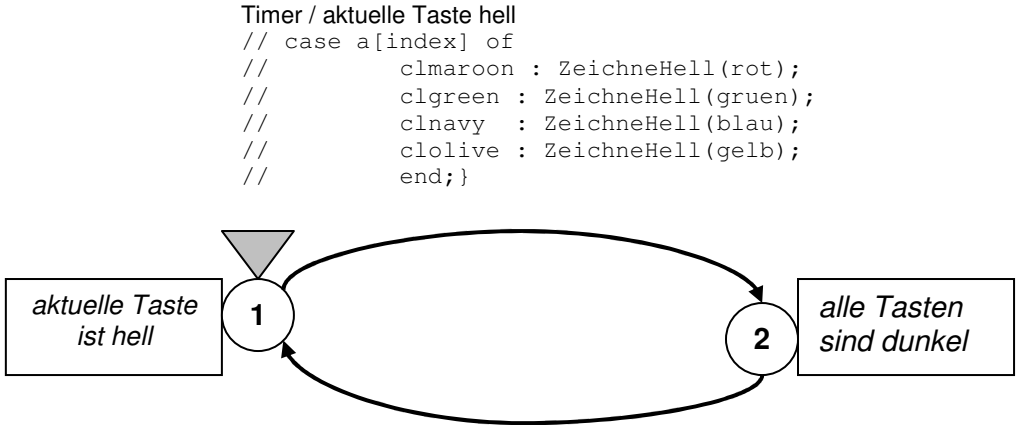
b) Die Zufallsbelegung soll in der Button-Click Methode erfolgen. Implementiere!

Aufgabe 2
Lass die ersten 10 Farben nacheinander leuchten.

a) Beschreibe die Funktion des nachstehende Zustandsmodells.

b) Wozu dient die Variable index?

c) Setze dazu das nachstehende Zustandsmodell als Zustandstafel um!



Aufgabe 3
Um Spielereingaben zu ermöglichen, wird die Prozedur **onMouseDown** des Formulars implementiert. Über *Pixel* kann die Farbe des Pixels unter dem Mauszeiger beim Tastendruck ermittelt werden.

```
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var Farbe : TColor;
begin
Farbe := canvas.Pixels[x,y];
[…]
end;
```

Setze das folgende Zustandsdiagramm in eine Zustandstabelle um und implementiere den algorithmischen Kern.

Zustandmodell