Input-Output

Aufgabe 3

Farbmischer-Spiel

Wie man den Benutzer mit passenden Hinweisen durch ein Programm führen kann, wird hier an einem kleinen Spielprogramm gezeigt, dem Farbmischer. Es geht darum, einen zufällig gewählten Farbton möglichst genau zu treffen. Machen Sie sich zunächst mit dem RGB-Farbraum vertraut. Wählen Sie in Processing *Tools/Farbauswahl.* Achten Sie dabei nur auf die Felder R,G und B auf der linken Seite. R steht für rot, G für grün und B für blau. Die Werte können jeweils zwischen 0 und 255 liegen. Es gibt daher 16.777.216 verschiedene Farben. Im Feld darunter ist der Farbwert hexadezimal ausgedrückt. Man erkennt das an der Raute links. Wir verwenden zur Erkennung von Hexadezimalzahlen den Index h.

$20AAC4_h$ beispielsweise bedeutet

R(ot) = 20_h = 2*16+0 = 32

G(rün) = AA_h = 10*16+10 = 170

B(lau) = $C4_h$= 12*16+4 = 196.

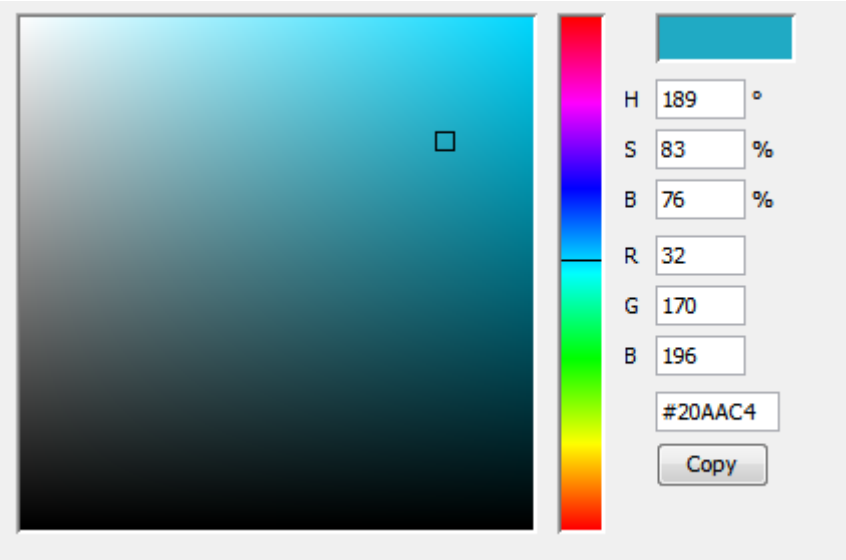


Abbildung 1

Der Spieler stellt durch Mausklick eine Zufallsfarbe her (siehe Abbildung 2, links). Sie wird in einem kleinen Quadrat oben links dargestellt. Oben rechts ist ein schwarzes Quadrat, dessen Farbe vom Spieler durch bestimmte Tastenkombinationen verändert werden kann.

Durch die Enter-Taste wird die Auswertung eingeleitet, die zu einer Punktzahl zwischen 0 und 100 führt.

Das Spiel Farbmischer wird 4 Zustände kennen. Zustand 2 ist in Abbildung 2 (rechts) zu sehen.

Die verschiedenen Textausgaben werden in Methoden gefasst. *text1Ausgeben()* enthält die Überschrift und die erste Anweisung an den Spieler. Ein schwarzes Quadrat wird schließlich noch oben links gezeichnet, dessen Farbe der Spieler später verändern soll. Dieser Zustand wird im setup() erzeugt:

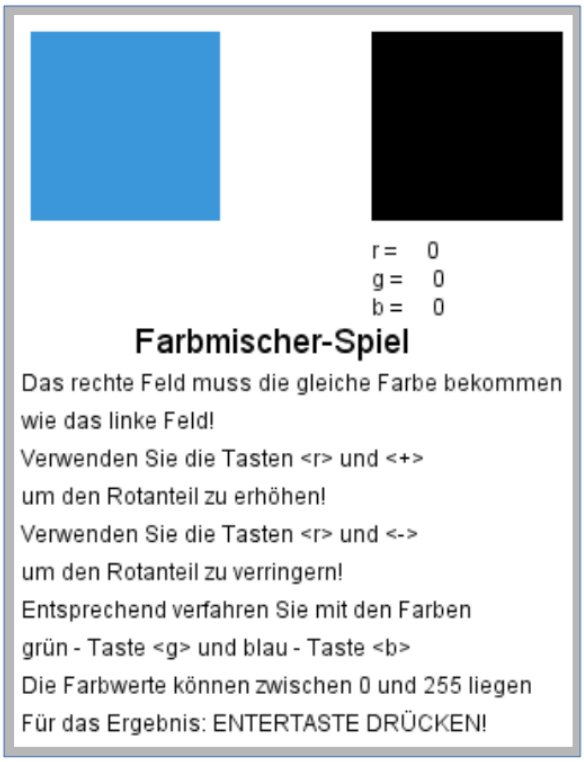


Abbildung 2

```
size(300, 390);
background(255);//weiß
font1 = loadFont("ArialMT-18.vlw");
font2 = loadFont("ArialMT-12.vlw");
red=0; green = 0; blue = 0;//schwarz
fill(red,green,blue);
rect(190,10,100,100);
text0Ausgeben();//r,g,b Werte anzeigen
text1Ausgeben();//Erläuternungen
```

Vergessen Sie nicht, die benötigten Attribute, wie red, green etc. zu deklarieren!

Sehen wir uns nun die Methode text0Ausgeben() an:

```
void text0Ausgeben() {
  fill(255);//weiß
  rect(190,110,110,50);//alte Farbwerte löschen
  fill(0);//schwarz
  textFont(font2);//kleine Schrift
  text("r =     " + str(red),190,130);//aktueller r-Wert
  text("g =     " + str(green), 190,145);
  text("b =     " + str(blue),190,160);
}
```

Diese Methode wird auch später sehr oft gebraucht werden. Immer dann nämlich, wenn der Spieler die Farbe des rechten Quadrats verändert hat.

Zustand 2 wird durch die Methode *mousePressed()* erzeugt. Dabei soll ein Quadrat links vom schwarzen Quadrat mit einer Zufallsfarbe gefüllt gezeichnet werden.

```
void mousePressed() {
  //rot-, gruen- und blau-Werte zwischen 0 und 255:
  rot = random(255);gruen = random(255); blau = random(255);
  rect(10,10,100,100);//Quadrat links
  fill(rot,gruen,blau);//wird mit Zufallsfarbe gefüllt
  fill(0);// Farbe auf schwarz zurückstellen
  text2Ausgeben(); //erläuternder Text siehe Abbildung Farbmischer12} b)
}
```

In *text2Ausgeben()* wird erklärt, wie der Spieler die Farben verändern kann und wie er die Auswertung in Punkten veranlassen kann (Enter-Taste). In Abbildung 3 ist ein Vorschlag für die Gestaltung des Hinweises für den Spieler abgebildet. Versuchen Sie, die unvollständige Methode *text2Ausgeben() fertig zu schreiben!*

```
void text2Ausgeben() {
  fill(255);
  noStroke();
```

```
  rect(0,185,300,21);//text1 löschen
  fill(0);
  textFont(font2);
  text("Das rechte Feld muss die gleiche Farbe bekommen",5,200);
  text("wie das linke Feld!",5,220);
  .....
}
```

Der Spieler soll die Farben rot, grün und blau des rechten Quadrats durch Tastenkombinationen regulieren können, um die Farbe des linken Quadrats möglichst genau zu treffen. Den Wert für rot beispielsweise soll er durch gleichzeitiges Drücken der beiden Tasten <r> und <+> erhöhen können. Wie im letzten Kapitel dargestellt, ist für die Auswertung eines Tastendrucks die Methode *keyPressed()* und für die Auswertung des Loslassens einer Taste die Methode *keyReleased()* zuständig.

Was allerdings noch offen ist: Wie schafft man es, dass die obigen Methoden nur auf das Drücken von gleichzeitig zwei Tasten reagieren? Damit beispielsweise eine noch zu schreibende Methode *rotErhoehen()* genau dann ausgelöst wird, wenn die Tasten <r> und <+> gleichzeitig gedrückt worden sind, bieten sich boolsche Attribute an. Diese sind wahr, wenn die zugeordnete Taste gedrückt ist. Falsch, wenn sie nicht gedrückt ist. Weil wir aber gleich fünf derartige Werte benötigen, verwenden wir einfacher ein

Array boolean[] keys mit fünf Elementen.

void setup() muss für die Erzeugung und Initialisierung des Arrays ergänzt werden. Dabei legt man zum Beispiel fest, dass keys[0]=false bedeutet, dass die <r>-Taste nicht gedrückt ist etc.
Ergänzung *setup()*:

```
.............
  keys=new boolean[5];
  keys[0]=false;keys[1]=false;keys[2]=false;
  keys[3]=false;keys[4]=false;
.....................
```

Bei jedem Durchgang der *draw()*-Methode} muss geprüft werden, ob zwei passende Tasten gedrückt sind. Das kann dann so aussehen:

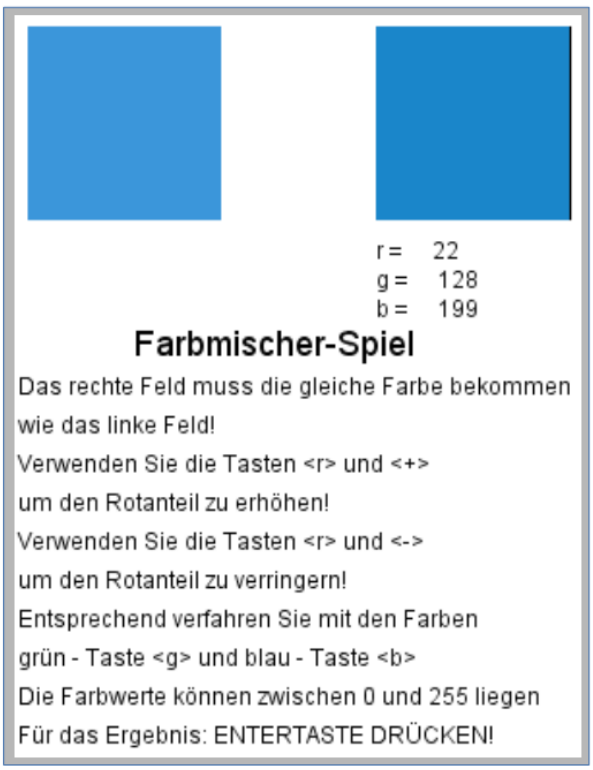
```
void draw() {
 if(keys[0] \&\& keys[3])  rotErhoehen();
 .................
}
```

Man erkennt am Namen der Methode *rotErhoehen()*, dass hier key[3]} wahr ist, wenn die <+>-Taste gedrückt ist. Insgesamt wird *rotErhoehen()* also durchgeführt, wenn die <r>- und die <+>-Tasten gedrückt sind. Ergänzen Sie zur Übung die fehlenden Zeilen!

Die benötigten Methoden zum Erhöhen oder Verringern eines Farbwertes sind schnell erstellt. So kann man beispielsweise die Methode rotErhoehen} gestalten:

```
void rotErhoehen() {
  if (red == 255) red =-1;//Höhere Werte als 255 gibt es nicht!
  red = red +1;
  fill(red,green,blue);
  rect(190,10,100,100); //Die Farbe wird angepasst
  text0Ausgeben();//Die neuen Farbwerte werden angezeigt
}
```

Entsprechend werden die fünf übrigen Methoden geschrieben. Vergessen Sie nicht, dass auch das Loslassen der Tasten registriert werden muss, damit z.B.
key[0]= false gesetzt wird, sobald man die <r>-Taste loslässt. Die *keyReleased()* Methode muss noch entsprechend der *keyPressed()* Methode geschrieben werden. Eine gute Übung!
Nun zu den Zuständen 3 und 4:



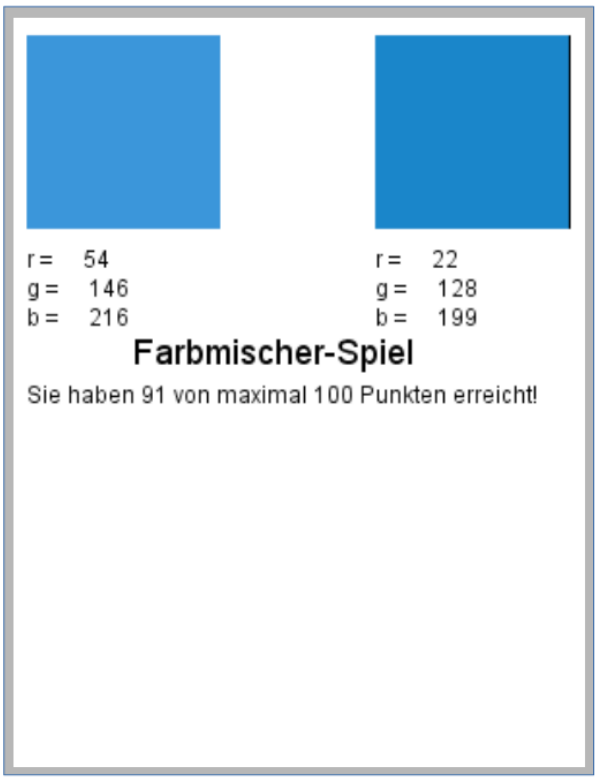


Abbildung 3

Spielzustand 3 zeigt lediglich einen Zustand, bei dem der Spieler vielleicht zufrieden mit der hergestellten Farbe ist. Er drückt daher auf die Entertaste und erwartet die Auswertung. Wir schreiben diese in eine Methode namens *ergebnisAnzeigen()*. Ausgewertet wird der Druck auf die Entertaste in der keyPressed() Methode durch:

```
if(keys == ENTER) ergebnisAnzeigen();
```

Wir erwarten von ergebnisAnzeigen(), dass, wie in Abbildung Farbmischer34} b), (dem 4.Zustand des Spiels) unter dem linken Quadrat die korrekten Farbwerte angezeigt werden und zusätzlich, nach dem Löschen des Textes, eine Punktwertung genannt wird:

```
void ergebnisAnzeigen() {
  fill(255);noStroke();
  rect(0,185,300,200);//text2 löschen
  fill(0); textFont(font2);
  text("r =     " + str(round(rot)),10,130);
  text("g =     " + str(round(gruen)), 10,145);
  text("b =     " + str(round(blau)),10,160);
  punkteAusgeben();
}
```

Die Methode *punkteAusgeben()* kann auf vielerlei Arten geschrieben werden. Überlegen Sie sich zunächst, wie eine gerechte Punktevergabe aussehen könnte. Welche Rechungen sind durchzuführen? Versuchen Sie es zunächst alleine. Sehr gut möglich, dass Ihr Wertung besser ist, als die in den Lösungen...