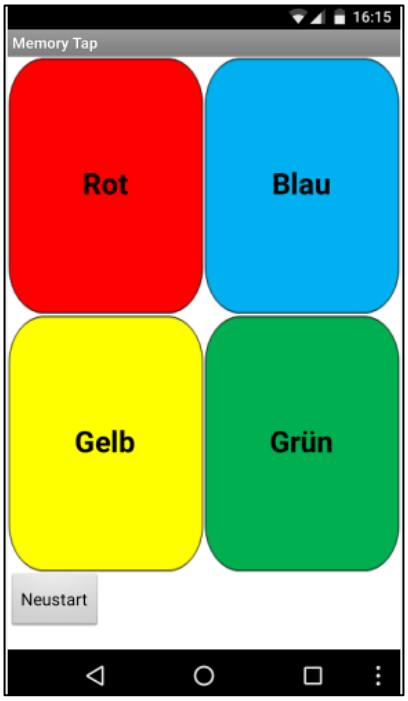
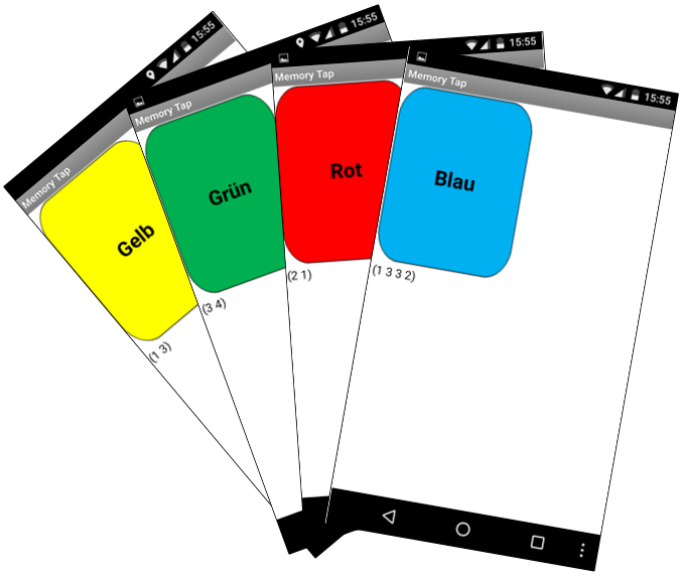
Memory Tap

Ein Spiel für Gedächtniskünstler

Worum geht es? Unsere App soll das Gedächtnis trainieren. Ein Kästchen blinkt mehrmals hintereinander in wechselnden Farben (z.B. rot-rot-blau-gelb). Wer es schafft, sich die Reihenfolge der Farben zu merken und durch Anklicken in einer Farbpalette wiederzugeben, kommt eine Runde weiter. Macht man einen Fehler, so ist das Spiel zu Ende. Wer die meisten Runden überlebt, hat gewonnen.





0. Unser Plan

Auch (und gerade) beim Programmieren ist es empfehlenswert, sich vorher einen Plan zu machen, wie man vorgeht. Dazu braucht man noch gar keine Programmierkenntnisse, einfach nur ein bisschen System und Köpfchen. Wir erkennen bei dem Spiel folgende Schritte, um die sich unser Programm kümmern muss:

1. *Die Farbpalette wird angezeigt.*

   Vor dem Spielstart sollen die vier Farben Rot, Blau, Gelb, Grün so angezeigt werden wie auf dem Bild oben. Erst, wenn die *Neustart*-Taste gedrückt wird, geht das Spiel los.

2. *Eine Farben-Folge aus zufällig gewählten Farben wird angezeigt (es blinkt vier Mal):*

   Vier Mal hintereinander wird eine (aus unseren vier Farben zufällig ausgewählte) Farbe für jeweils eine Sekunde angezeigt. Die Reihenfolge der angezeigten Farben müssen wir uns für später merken. Wir schreiben bei jeder Farbanzeige die Farbenzahl (1, 2, 3 oder 4) in eine Liste (*Farbenliste*). Nachdem vier Farben geblinkt haben (*Blinkanzahl* = 4), dann stoppt das Blinken und der Bildschirm zeigt wieder die Farbpalette mit allen Farben.

3. *Der Spieler klickt die Farben (möglichst) in der Blink-Reihenfolge an:*

   Vier Mal klickt der Spieler nun auf Farb-Tasten in der Palette. Wir müssen vergleichen, ob seine Klick-Reihenfolge mit der Reihenfolge in der Farbenliste übereinstimmt. Beim ersten Klick wird nachgeschaut, ob die geklickte Farbe dieselbe ist wie der erste Eintrag in unserer Farbenliste. Wenn ja, dann löschen wir den ersten Eintrag einfach. Damit erreichen wir, dass der nächste Klick auch wieder mit dem ersten Eintrag verglichen werden muss, denn die übrigbleibenden Farben rutschen automatisch in der Liste einen Platz nach vorne. Wenn richtig geraten wird, dann wird die Liste jedes Mal um einen Eintrag kürzer. Beim letzten richtigen Klick wird der letzte Eintrag aus der Farbenliste gelöscht. Wird ein Fehler gemacht, dann wird „Game Over“ gezeigt und die App bleibt stehen.

4. *Wenn alles stimmt, beginnt die nächste Runde: Es blinken wieder vier Farben*

   Wenn der Spieler alles richtig gemacht hat, ist die Farbenliste jetzt leer. Der Spieler kann die nächste Runde spielen. Also setzen wir bei leerer *Farbenliste* die *Blinkanzahl* wieder zurück auf 0, denn dann beginnt das Blinken wieder von vorne.

1. Die Farbpalette

a) Lege ein neues Projekt an, nenne es *MemoryTap*.

b) Wir brauchen vier verschiedenfarbige Felder, auf die man tippen kann. Ziehe links aus der Palette vier *Tasten* auf die Bildschirmfläche.

c) Rechts unter *Eigenschaften* können jeweils die Eigenschaften der Tasten geändert werden, wie zum Beispiel hier für die rote Taste:

   - Setze die Hintergrundfarbe auf *Rot*.
   - Mache die Schrift **fett** (Häkchen bei *Fettschrift*).
   - Setze die Schriftgröße auf 24.
   - Ändere den Text von „Text für Taste1“ zu „Rot“.

d) Füge die vier farbigen Felder als geschlossene Fläche in einer *Tabelle* zusammen:
   - Ziehe *Tabelle* aus *Palette* | *Anordnung* auf den Screen und schiebe die vier Tasten dort hinein.
   - Setze für jede Taste die Breite auf *50%* und die Höhe auf *50%*.

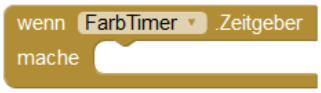
2. Eine Farb-Reihenfolge mit Zufallsfarben wird angezeigt

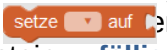
Unsere App soll nun so erweitert werden, dass sie zunächst eine Reihenfolge von vier zufällig gewählten Farben durch vier Mal Blinken vorgibt. Für das Aufleuchten von vier Farben nacheinander für jeweils eine Sekunde brauchen wir einen *Timer*.

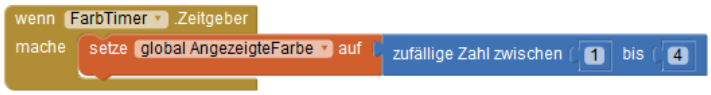
a) Wechsle zum *Design-Editor*. Ziehe eine *Uhr* aus der Palette (unter *Sensoren*) auf die Benutzeroberfläche und nenne sie *FarbTimer*.

b) Wechsle zum *Blöcke*-Editor. Erstelle eine Variable (*Eingebaut* → *Variablen*) und nenne sie *AngezeigteFarbe*. Diese soll eine Zahl zwischen 1 und 4 für die zufällig ausgewählte Farbe speichern. Gib ihr den Startwert 1.

c) Wähle in der Palette links den *FarbTimer* aus. Ziehe den Block wenn **FarbTimer.Zeitgeber** auf die Arbeitsfläche. Alles in diesem Block wird standardmäßig alle 1000 ms (= 1 Mal pro Sekunde) ausgeführt – perfekt für unseren Farbenblinker.

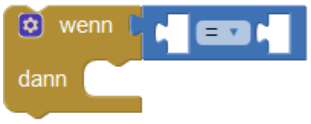


d) Jede Sekunde soll nun eine zufällig ausgewählte Farbe angezeigt werden. Wir setzen daher als erstes unsere Variable *AngezeigteFarbe* auf eine zufällige Zahl zwischen 1 und 4: Klicke dazu auf *Eingebaut* | *Variablen* und ziehe  en Arbeitsbereich. Wähle *global AngezeigteFarbe* aus. Den Zufallszahlen-Baustein **zufällige Zahl zwischen.. bis..** findest du unter *Eingebaut* | *Mathematik*. Setze nun den Wert *AngezeigteFarbe* auf eine Zufallszahl zwischen 1 und 4. Das Ganze soll jede Sekunde neu passieren; es wird also in unseren *FarbTimer*-Baustein eingebaut.



e) Den Wert der Variablen *AngezeigteFarbe* kannst du über *Eingebaut* | *Variablen* | auslesen. Je nach Wert soll nun für eine unserer vier Farben die Sichtbarkeit eingeschaltet werden (*Sichtbar* wird auf *wahr* gesetzt), während für die anderen drei Farben die Sichtbarkeit ausgeschaltet werden soll (*Sichtbar* wird *falsch* gesetzt). Wir brauchen also eine *Wenn..Dann..*-Bedingung. Diesen Block findest du unter *Steuerung* | **wenn..dann..**

f) Die Bedingung ist ein neuer Block hinter dem Wort ***wenn***. Hier fragen wir ab, ob die Zahl in *AngezeigteFarbe* eine 1 ist. In diesem Fall soll Rot angezeigt werden. Benutze für den Vergleich den blauen **=**-Baustein aus *Eingebaut* | *Mathematik* und fülle die zu vergleichenden Werte mit der Variablen *AngezeigteFarbe* und der Zahl 1.



g) Wenn der Vergleich stimmt, dann werden die Blöcke im hinter dem Wort ***dann*** ausgeführt. Hier musst du also die Sichtbarkeit für die Farbe Rot einschalten und für alle anderen Farben ausschalten.

h) Ergänze drei weitere **wenn..dann..**-Blöcke für die anderen drei Farben und setze alle vier **wenn..dann..**-Blöcke in den *FarbTimer*-Block hinter die Belegung der Variablen *AngezeigteFarbe* mit einer Zufallszahl. Nun wird jede Sekunde eine neue Farbe angezeigt.

i) Teste dein Programm. Weil der Timer ewig läuft, werden die Farben nun ewig blinken.

3. Beschränke die Farb-Anzahl beim Blinken auf vier Farben

a) Damit das Blinken nach vier Sekunden (bzw. vier Farben) aufhört, brauchen wir einen Zähler. Erstelle eine Variable *Blinkzähler* und belege sie mit 1.

b) Nachdem eine Farbe angezeigt wurde (also einmal pro Sekunde), soll sich die Zahl im Blinkzähler um eins erhöhen:

- Klicke unter *Eingebaut* auf *Mathematik* und wähle den **+**-Block aus.
- Füge die beiden Summanden dort ein: den Wert in Blinkzähler und die Zahl 1.
- Setze den (neuen) Wert von Blinkzähler auf das Ergebnis dieser Summe.
- Setze diesen Block ans Ende innerhalb des *FarbTimer*-Blocks. Nun erhöht sich nach jedem Blinken die Zahl des Blinkzählers um eins.

c) Entscheide mit Hilfe eines **wenn..dann..**-Blocks, ob die Anweisungen im *FarbTimer*-Block überhaupt ausgeführt werden sollen (nämlich nur, wenn unser neuer *Blinkzähler*-Wert kleiner oder gleich 4 ist, wir also im Blinkmodus sind).

d) Verschiebe die vier **wenn..dann..**-Blöcke aus 2.h) in den ***dann*** -Teil vom Blinkmodus (3. c). Teste die App. Jetzt blinken genau vier Farben!

4. Merke dir die aktuelle Farb-Reihenfolge in einer Liste

Damit wir die angezeigten Farben und ihre Reihenfolge später mit der Eingabe des Spielers vergleichen können, müssen wir uns die Farben merken. Am besten, wir schreiben sie als Liste auf: Der erste Eintrag in dieser Liste ist die erste gezeigte Farbe, der zweite Eintrag ist die zweite Farbe und so weiter.

a) Erstelle eine neue Variable *Farbenliste*. Belege sie mit einer leeren Liste, indem du den Baustein leere Liste erstellen aus *Eingebaut* | *Listen* verwendest.

b) Füge nun nach jeder Farbanzeige die angezeigte Farbe (Inhalt der Variablen *AngezeigteFarbe*) am Ende der *Farbenliste* ein. Verwende dazu den Listen-Baustein Elemente der Liste hinzufügen.

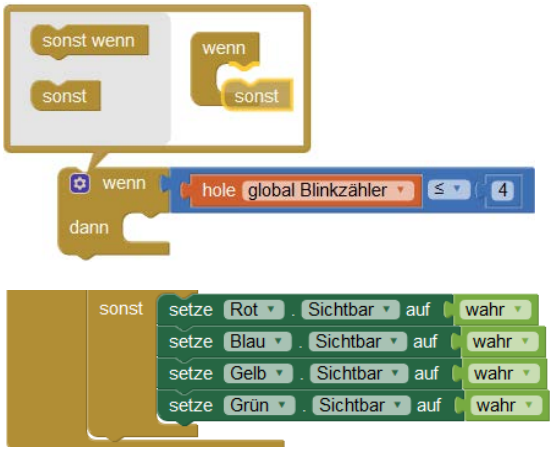
c) Mit einer neuen Bezeichnung können wir uns testweise den Inhalt der Liste anzeigen lassen. Erstelle im Designer eine Bezeichnung *FarblisteBezeichnung* und setze im Blöcke-Editor die Anzeige-Anweisung unter den Listenbaustein aus 4.b).

5. Zeige nach dem Blinken wieder die Farbpalette an

Nach Ende des Blinkens soll nun wieder die komplette Farbpalette zu sehen sein, also alle vier Tasten auf einmal. Das Ende des Blinkens ist erreicht, wenn unser Blinkzähler den Wert 4 oder höher hat. Bisher haben wir im Farbtimer im äußersten **wenn..dann..**-Block nur einen ***dann***-Teil, den Blink-Modus. Nun fügen wir einen Sonst-Fall hinzu, wo alle Anweisungen hinkommen, die nach Ende des Blinkens ausgeführt werden sollen (Tipp-Modus).

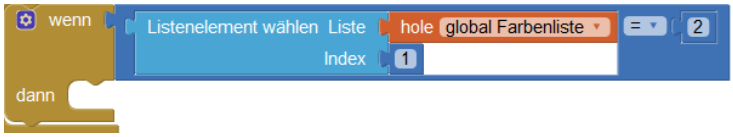
a) Klicke auf das blaue Erweiterungs-Symbol im äußersten **wenn..dann..**-Block und ziehe das **sonst**-Symbol aus dem grauen Bereich links unter das **wenn**-Symbol im weißen Bereich rechts. Die Bedingung wird dadurch unten durch einen Sonst-Fall erweitert.

b) Dort hinein kommen jetzt die vier Anweisungen, die die Sichtbarkeit aller unserer Farb-Tasten wieder auf *wahr* setzen. Durch die Abfrage, ob wir noch im Blink-Modus sind (Blinkzähler kleiner oder gleich vier), unterscheiden wir nun den Wenn-Fall (die Farben blinken nacheinander) vom Sonst-Fall (die gesamte Farbpalette wird angezeigt, das ist der Tipp-Modus).

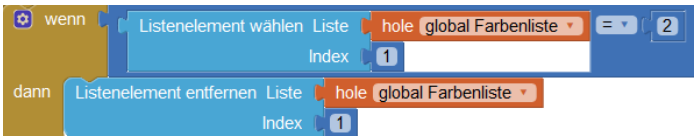


6. Vergleiche die Spieler-Klicks mit der gespeicherten Farbsequenz

a) Bei jedem Klick auf eine der Tasten muss nun die angeklickte Farbe mit dem entsprechenden Listenelement unserer *Farbenliste* verglichen werden. Für jede Farbtaste brauchen wir einen neuen Block: Im Block wenn **Blau.Klick** mache musst du also die aktuell erste Farbe in unserer Liste (die mit *index*-Nummer 1) mit Blau (Farbnummer 2) vergleichen. Verwende dazu den Listen-Baustein Listenelement wählen.



b) Wenn der Vergleich stimmt (Wenn-Teil), dann soll dieser erste Farbeintrag aus der Liste entfernt werden. Verwende dazu den Listen-Baustein Listenelement entfernen. Die *index*-Nummer des Listenelements musst du auf 1 setzen.



Nun ist die Liste um eine Farbe kürzer. Automatisch rücken die nachfolgenden Farben einen Listenplatz weiter nach vorne. Die bisher an zweiter Stelle stehende Farbe steht nun an erster Stelle, die dritte Farbe steht nun auf Platz 2 usw.

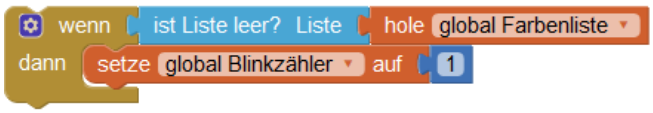
c) Füge nun in den drei anderen Blöcken wenn **Rot.Klick** mache, wenn **Gelb.Klick** mache und wenn **Grün.Klick** mache die entsprechenden Vergleiche hinzu. Hat der Spieler auf die richtige Farbe geklickt, dann löschst du jedes Mal die aktuelle Farbe vom ersten Listenplatz. Hat der Spieler vier Mal richtig geklickt, dann ist die Liste leer.

d) Achtung, im Fall einer leeren Liste darf der Vergleich zwischen einer Farbzahl und dem Wert auf dem ersten Listenplatz gar nicht stattfinden: Da es bei einer leeren Liste keinen ersten Listenplatz gibt, bekommst du sonst eine Fehlermeldung. Wir vermeiden diesen Fehler, indem wir den Vergleich nur dann machen, wenn die Liste noch nicht leer ist. Dafür brauchen wir wieder eine **wenn..-dann..**-Abfrage und das Ergebnis des Listen-Blocks ist Liste leer?. Die Bedingung ist, dass die Liste NICHT leer ist, also brauchen wir noch den Block Logik | nicht. Füge auch diese Abfrage den anderen drei Blöcken wenn **Rot.Klick** mache, wenn **Gelb.Klick** mache und wenn **Grün.Klick** mache hinzu.

7. Wenn alle Farben stimmen, lass eine neue Farbenreihe blinken

a) Wenn kein Fehler gemacht wurde, dann ist nach vier Klicks die Farbenliste leer. In diesem Fall wollen wir eine neue blinkende Farbsequenz anzeigen. Wir brauchen also wieder eine **wenn..-dann..**-Abfrage mit dem Ergebnis des Listen-Blocks ist Liste leer? als Bedingung (**wenn**-Teil). Wo platzieren wir diesen Block? Am besten dort, wo wir immer die Farbpalette anzeigen: im Sonst-Teil der äußeren **wenn..-dann..**-Abfrage in unserem *FarbTimer* (siehe Abschnitt 5b).

b) In den Wenn-Fall unserer Abfrage, ob die Liste leer ist, kommt nun die Anweisung, wieder mit dem Blinken zu beginnen. Was heißt das genau? Erinnere dich an unsere Variable *Blinkzähler* aus Abschnitt 3a. Das war die Zahl, die hochgezählt wurde, während die vier Farben nacheinander angezeigt wurden. Als der Wert in *Blinkzähler* die Zahl 5 erreicht hatte, wurde das Blinken beendet. Du musst also nichts weiter tun, als den *Blinkzähler* wieder auf 1 zu setzen; schon beginnt wieder das Blinken im Wenn-Teil vom *FarbTimer*-Block.



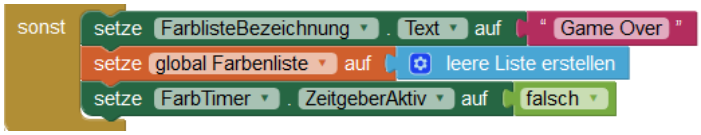
8. Beende das Spiel, sobald ein Fehler gemacht wird

Jetzt kann man schon richtig spielen: Farben blinken in zufälliger Reihenfolge, eine Farbauswahl wird durch Anklicken in der Farbpalette gemacht, und bei richtigem Anklicken beginnt das Spiel von vorne. Alles gut? Na ja, wenn der Spieler eine Farbe falsch anklickt, dann wird er bisher überhaupt nicht bestraft. In unserem jetzigen Spiel passiert dann gar nichts, es wird einfach gewartet, bis der Spieler es noch einmal probiert. Nach dem Motto, jeder hat eine zweite Chance verdient. Und eine dritte… und eine vierte…

Nein, das geht natürlich nicht. Wir werden das Spiel nun strenger machen und falsche Eingaben nicht mehr tolerieren. Wenn also auf eine Farbtaste geklickt wird, und dies ist NICHT die richtige Farbe (die auf dem aktuellen ersten Listenplatz), dann soll „Game Over“ angezeigt werden. Nun sollte allerdings alles andere einfrieren, damit das Spiel nicht mehr weitergeht. Wie bekommst du das hin?

a) Zunächst brauchen wir in den vier Blöcken wenn **Blau.Klick** mache, wenn **Gelb.Klick** mache, wenn **Grün.Klick** mache und wenn **Rot.Klick** mache bei den **wenn..-dann..**-Abfragen, ob richtig geklickt wurde, noch einen Sonst-Teil . Dort muss stehen, was passiert, wenn der aktuelle Klick NICHT die richtige Farbe ist. (Wie du einen **sonst**-Teil hinzufügst, steht übrigens in Abschnitt 5a).

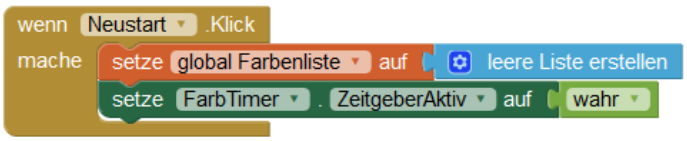
b) In den jeweiligen neuen **sonst**-Teil kommen nun die Aktionen hinein, die bei einem Fehler passieren: 1. Es erscheint der Text „Game Over“; dazu missbrauchen wir unsere Bezeichnung, wo wir bisher testweise den Inhalt der Liste angezeigt haben. 2. Unsere Farbenliste wird geleert, damit man nicht weiter tippen kann. 3. Wir deaktivieren unseren FarbTimer, damit es nicht wieder von neuem anfängt zu blinken (denn bei leerer Farbenliste setzen wir ja eigentlich unseren Blinkzähler auf 1 zurück, siehe Abschnitt 7b). Nun ist das Spiel zu Ende und der Spieler kann nichts mehr machen.



9. Neustart des Spiels

Es ist allerdings blöd, wenn man nochmal spielen möchte, die App jedes Mal zu schließen und das Spiel neu zu starten. Also bauen wir noch eine Neustart-Taste ein, die dazu führt, dass die Farben wieder neu zu blinken beginnen und man wieder klicken kann. Wie man im Designer eine Taste erstellt, weißt du ja schon. Nenne die Taste *Neustart*.

a) Im Block wenn **Neustart.Klick** mache muss stehen, was passiert, wenn die Neustarttaste geklickt wird: 1. Leere die Farbenliste (das passiert zwar nach einem Fehler automatisch, aber der Spieler könnte die Taste ja auch ohne Fehler klicken; dann stehen noch Farben in der Liste). 2. Aktiviere den Farbtimer (den haben wir nach einem Fehler deaktiviert).



Nun kann man immer wieder spielen, bis es zu langweilig wird. Um es schwieriger zu machen, kannst du die Anzahl der blinkenden Farben in jeder Runde automatisch erhöhen!