

Von der Aufgabe zum Code

Daniel Käs
Robert Lubkoll

Von der Aufgabe zum Code

- Im großen Rahmen: siehe Softwaretechnik
- Im kleinen Rahmen: Erfahrungssache

es folgt ein wenig Erfahrung

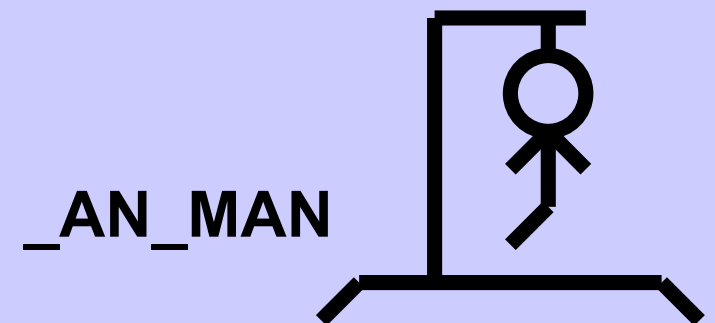
Unsere Themen

Womit fange ich eigentlich an?
Wie entwerfe ich ein Programm?
Wie behalte ich den Überblick?

Womit fange ich eigentlich an?

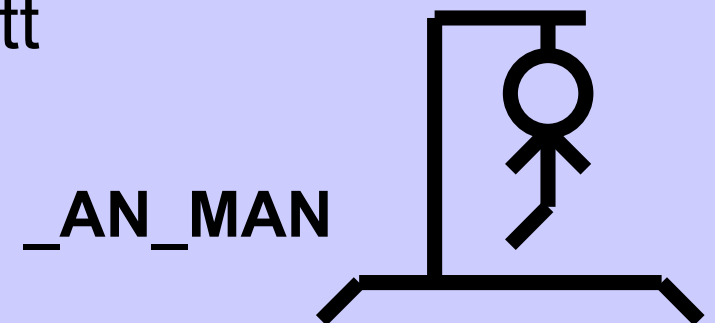
Beispielaufgabe

- Schreibe das Spiel Hangman für die Konsole.
- Es soll von einem Spieler beliebig oft spielbar sein.
- In jedem Spielschritt soll angezeigt werden:
 - die verbleibende Menge an Rateversuchen
 - die bereits errateten Buchstaben des Lösungswortes



1. Übersicht verschaffen

- Spielregeln Hangman
 - Ein Wort wird gewählt
 - Spieler muss Buchstaben des Wortes erraten
 - Position erratener Buchstaben wird angezeigt
 - falscher Rateversuch führt zu Hangman-Strich
 - gewonnen wenn Wort erraten
 - verloren wenn Hangman komplett



2. Ablauf skizzieren

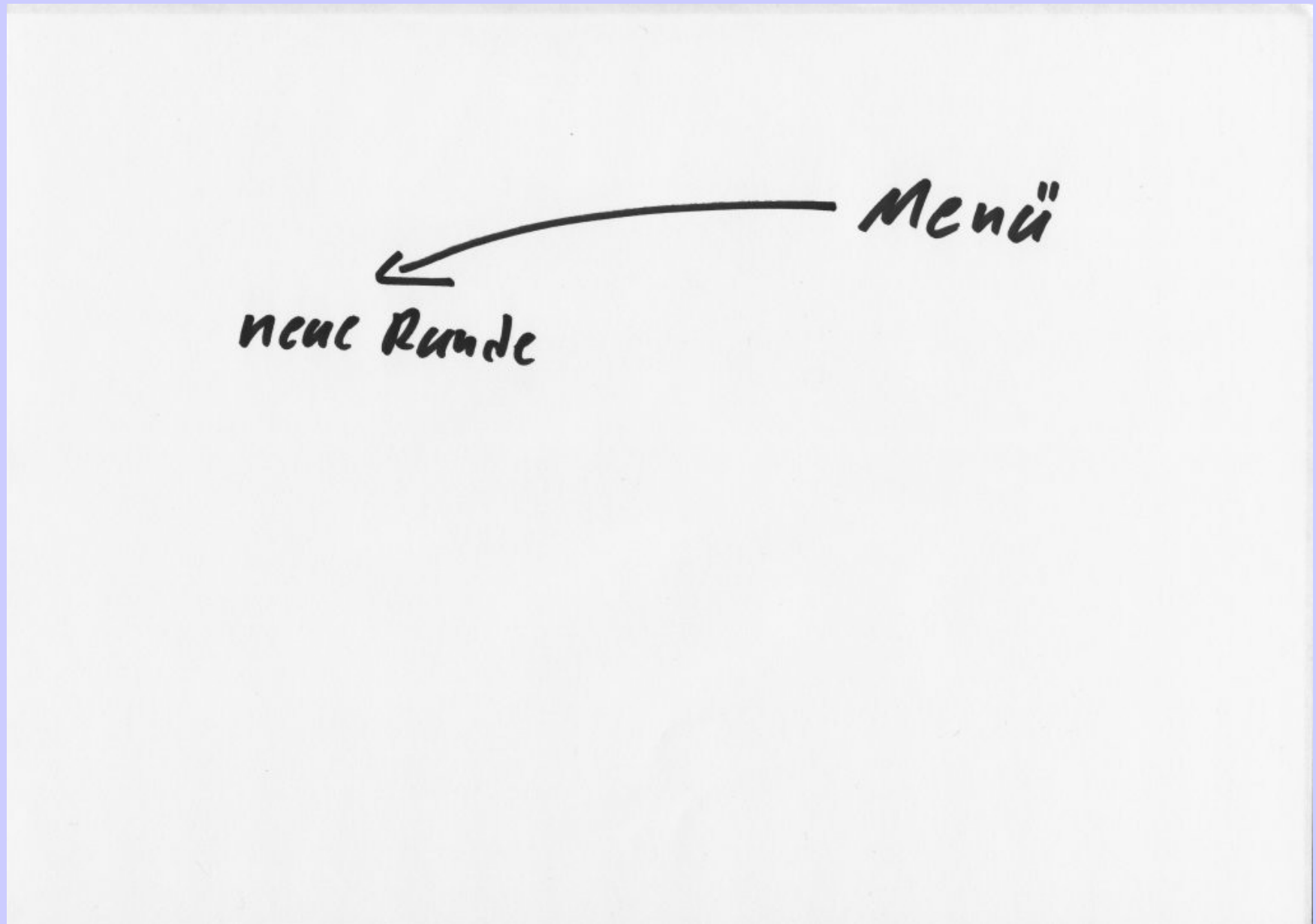
- Braindump
- soll Gedanken ordnen
- Aufgabe in Teilprobleme zerlegen

- Viele Probleme in einer erdachten Lösung entdeckt man erst wenn man sie aufschreibt

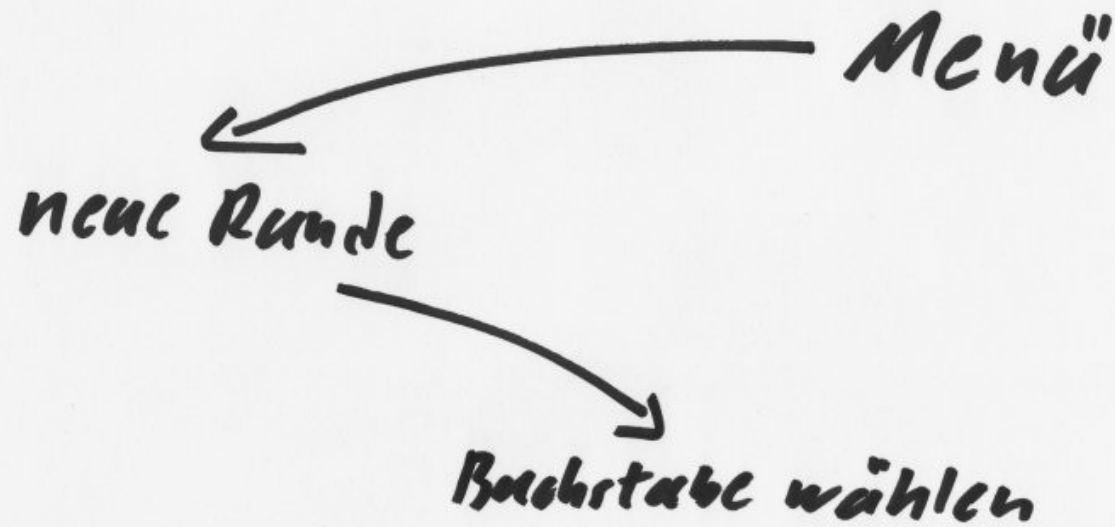
Beispiel einer Ablaufskizze



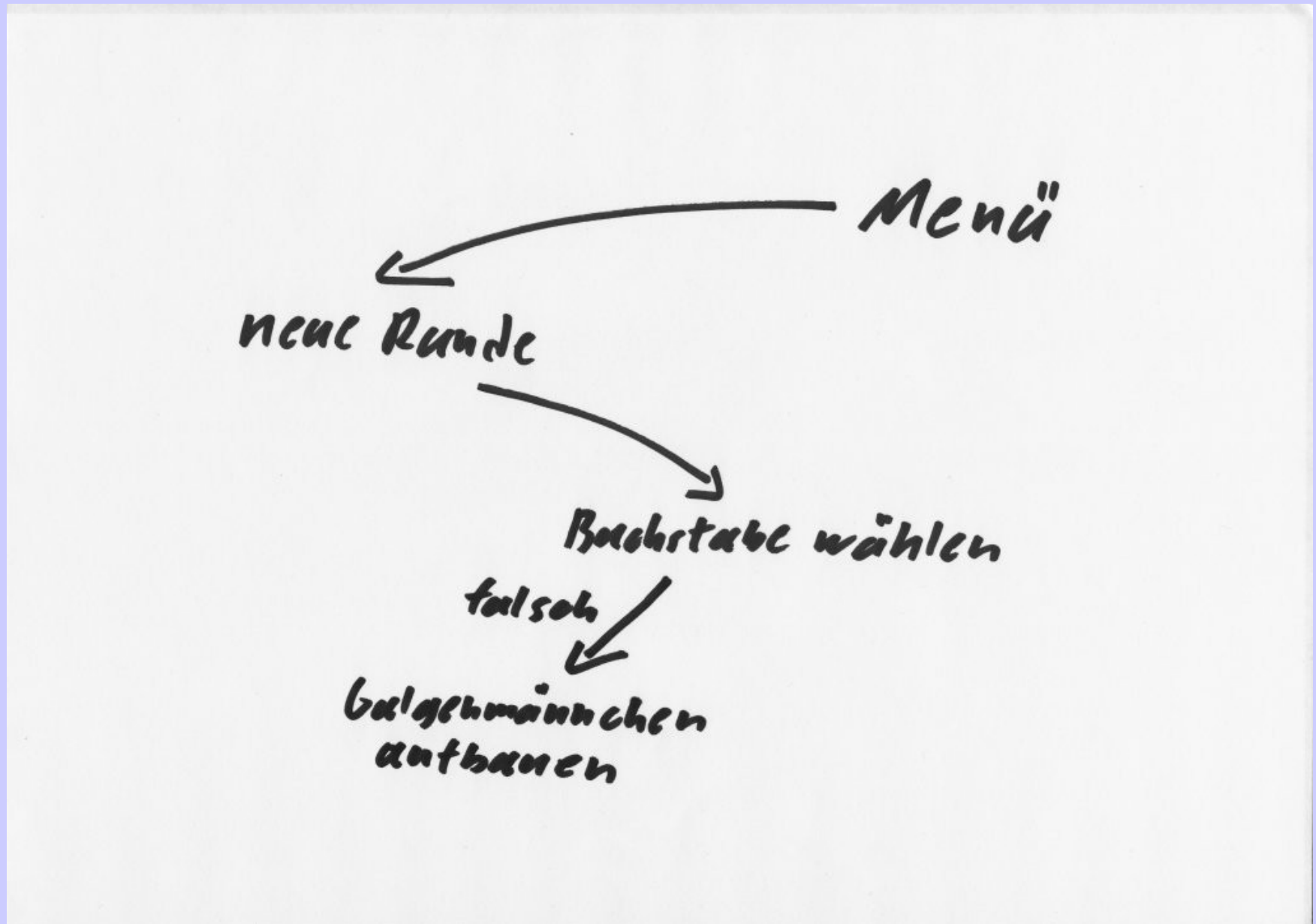
Beispiel einer Ablaufskizze



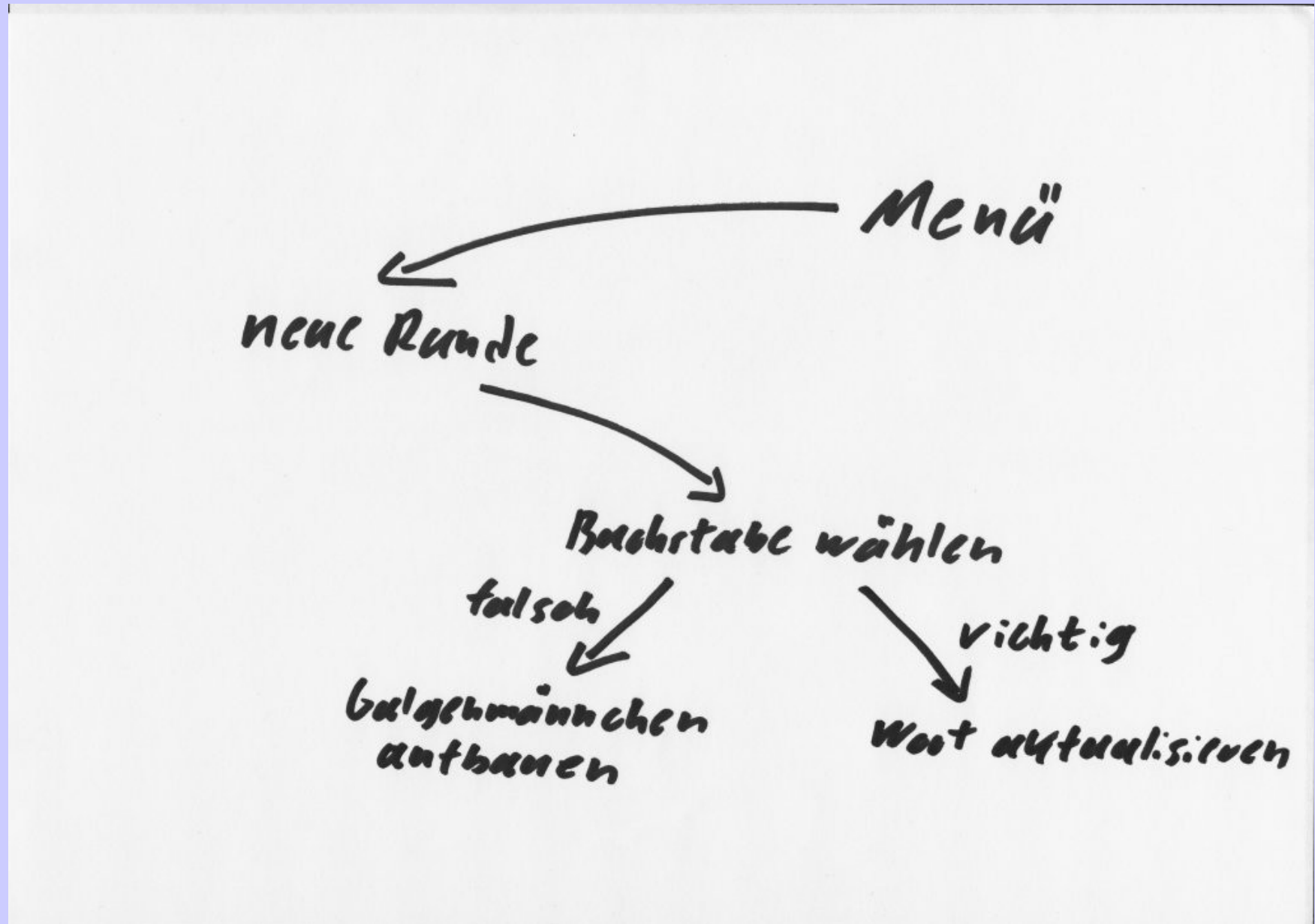
Beispiel einer Ablaufskizze



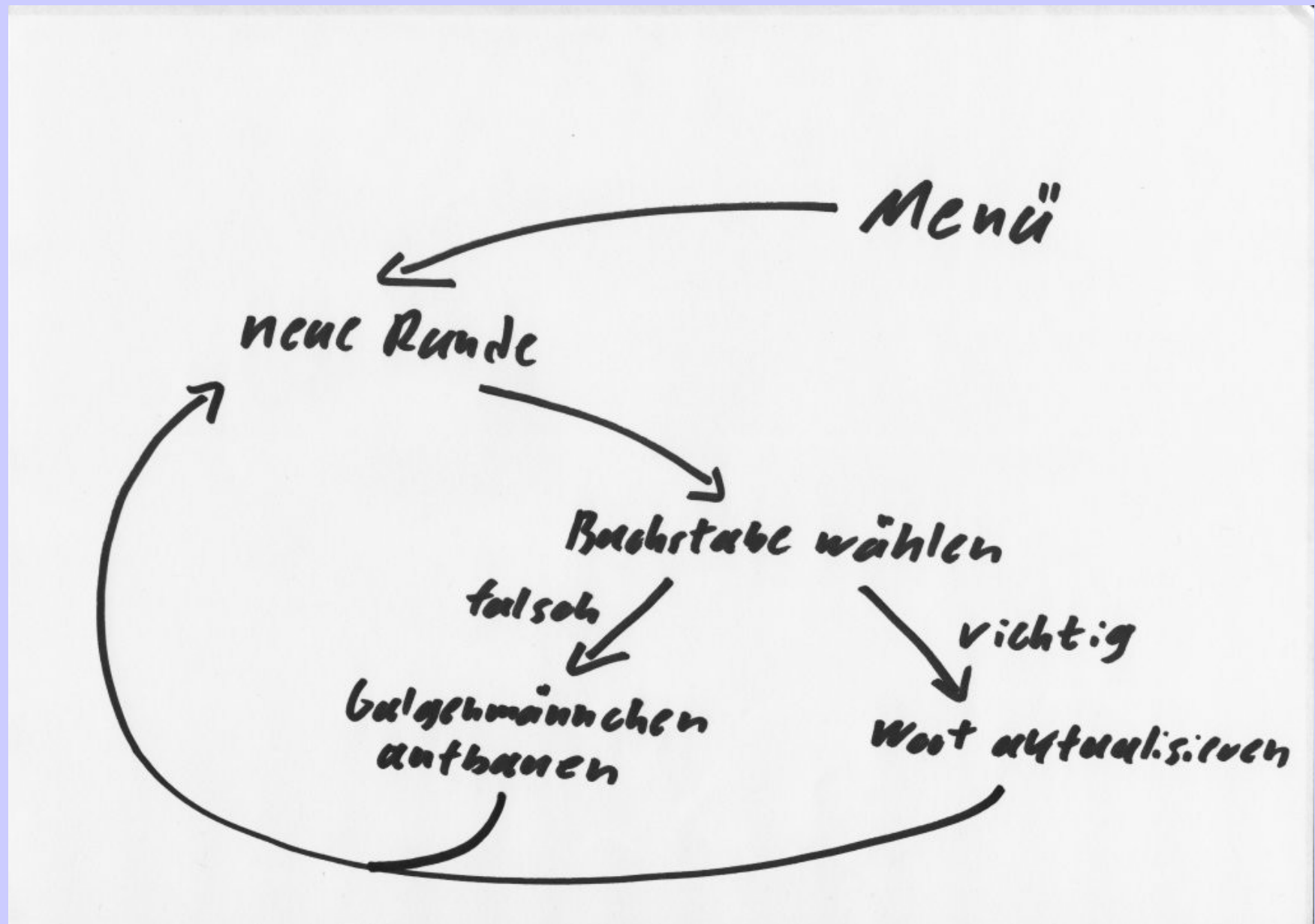
Beispiel einer Ablaufskizze



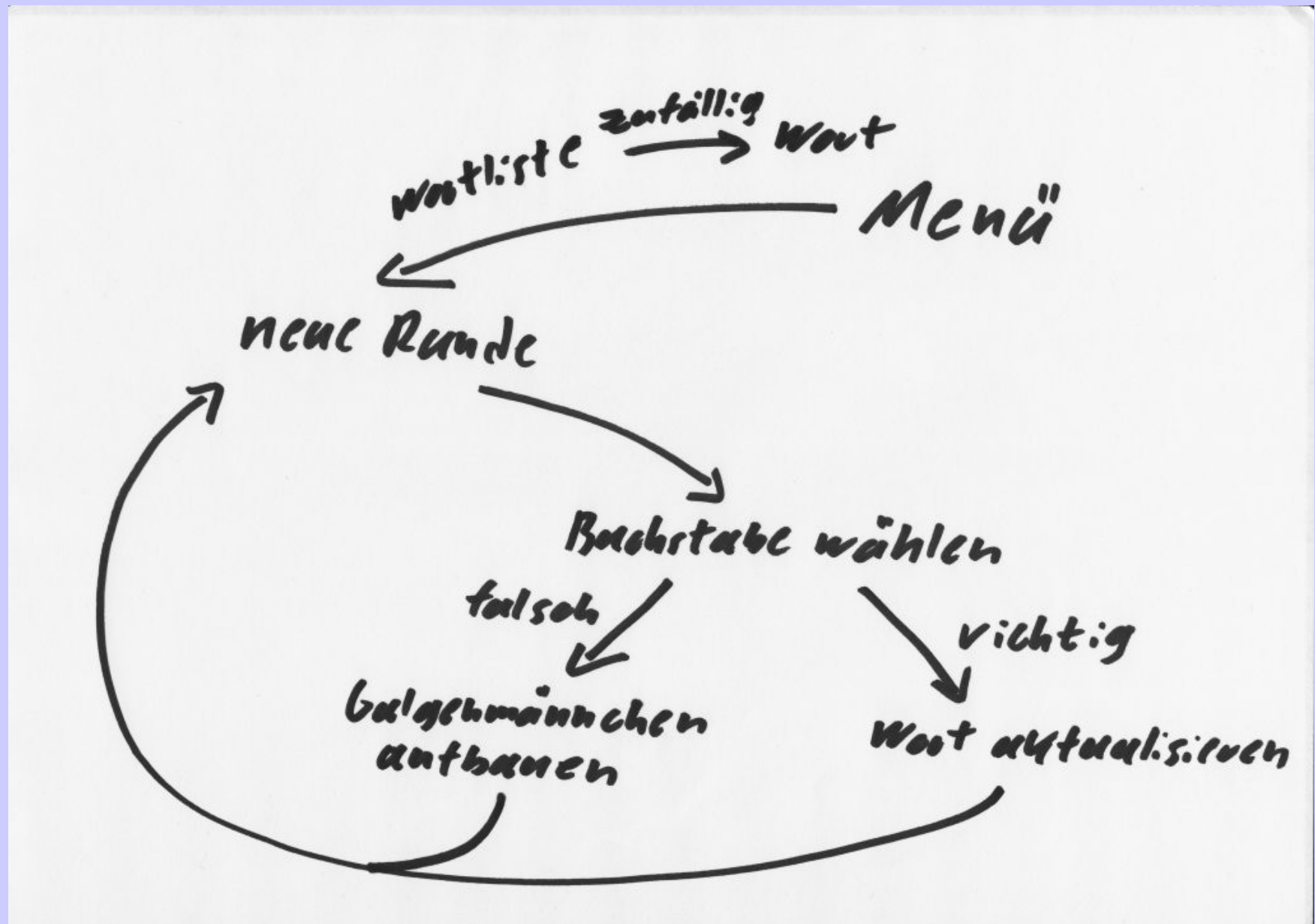
Beispiel einer Ablaufskizze



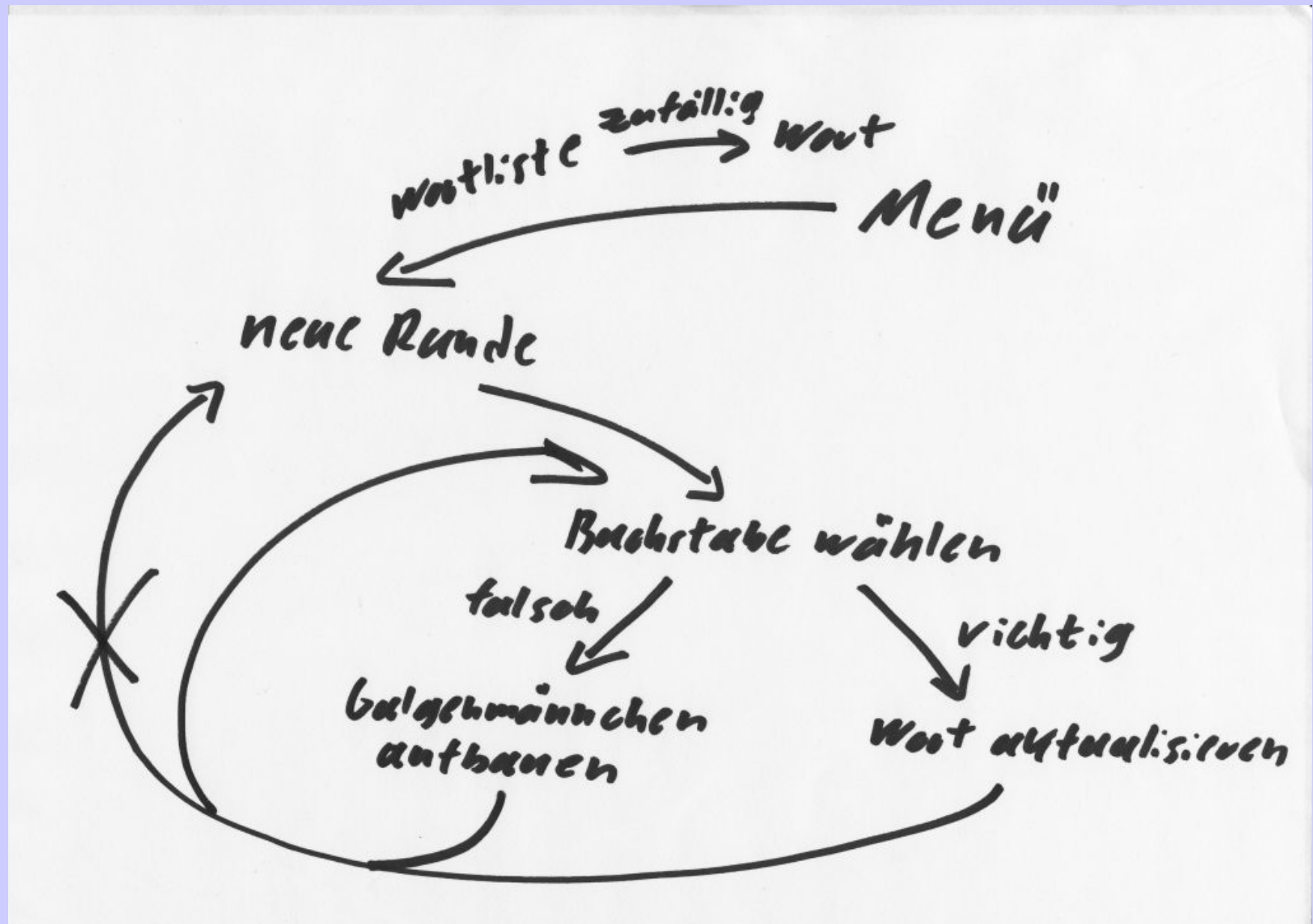
Beispiel einer Ablaufskizze



Beispiel einer Ablaufskizze



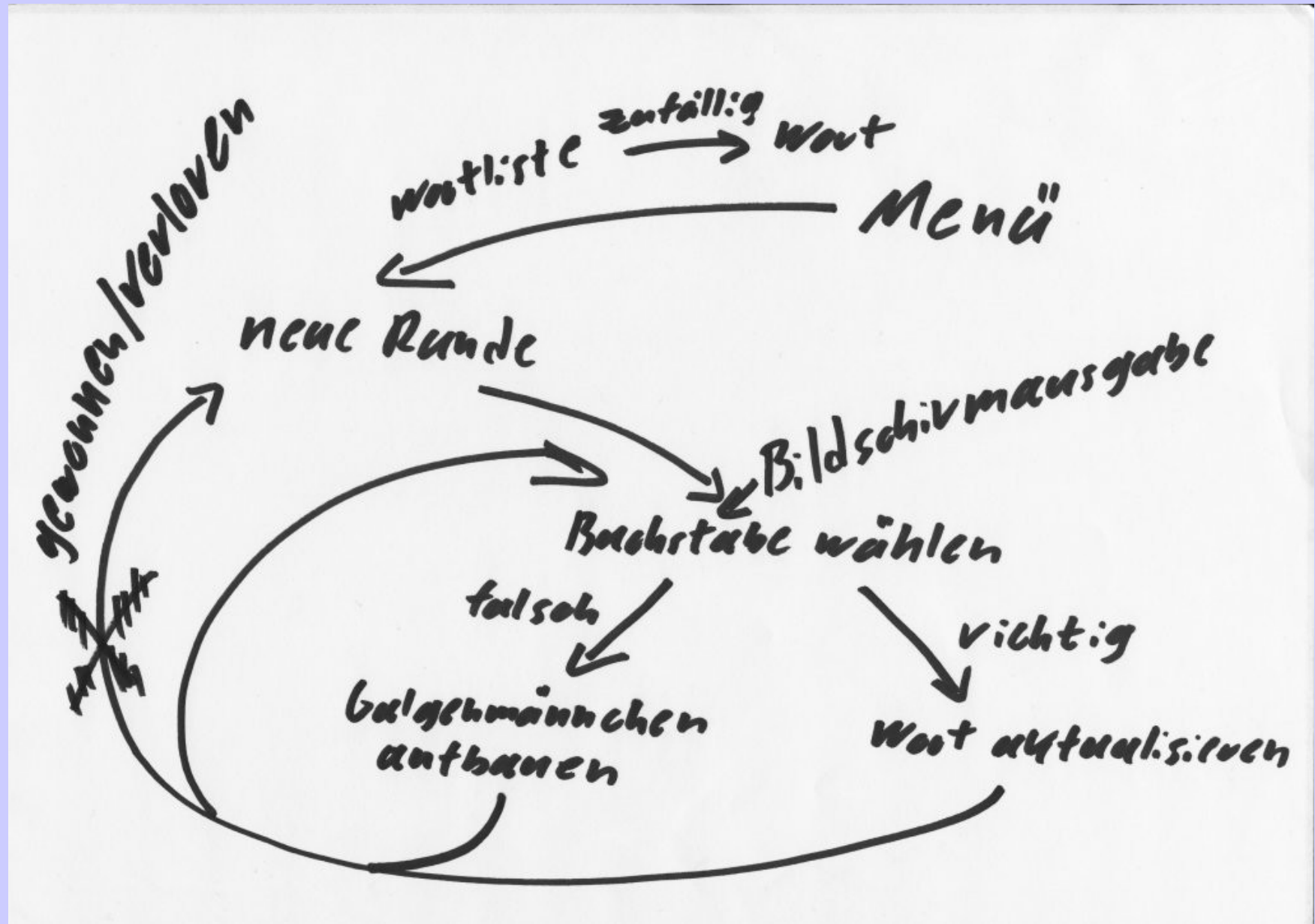
Beispiel einer Ablaufskizze



Beispiel einer Ablaufskizze



Beispiel einer Ablaufskizze



Ablaufskizzen sind individuell

Menüschleife {

Menü → Ende / neues Spiel
wähle Wort w

Spielschleife {

erfrage Buchstaben

verarbeitete Buchstaben → Suche B. im Wort
↳ Galgen
↳ B. anzeigen

}

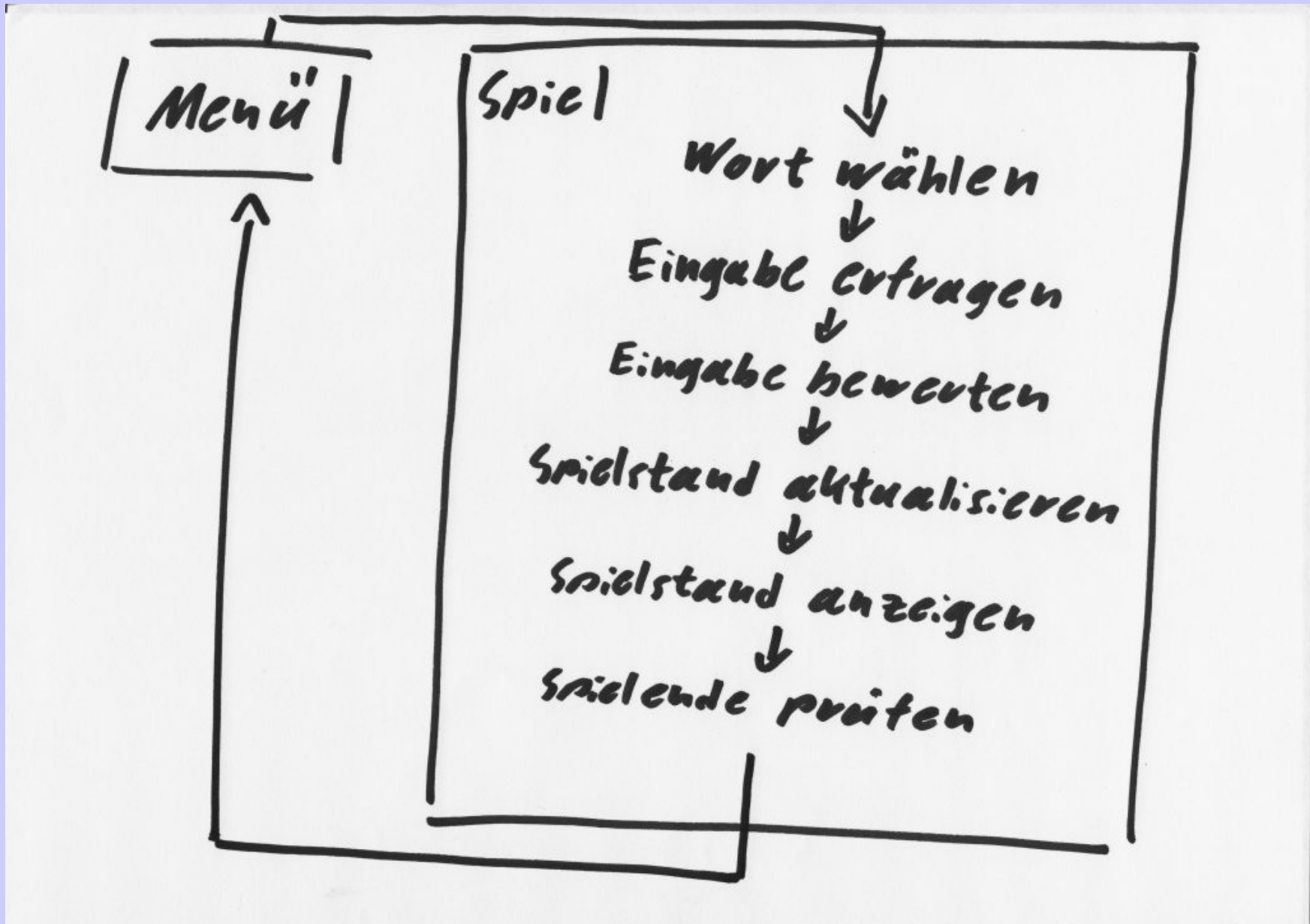
}

Wie entwerfe ich ein Programm?

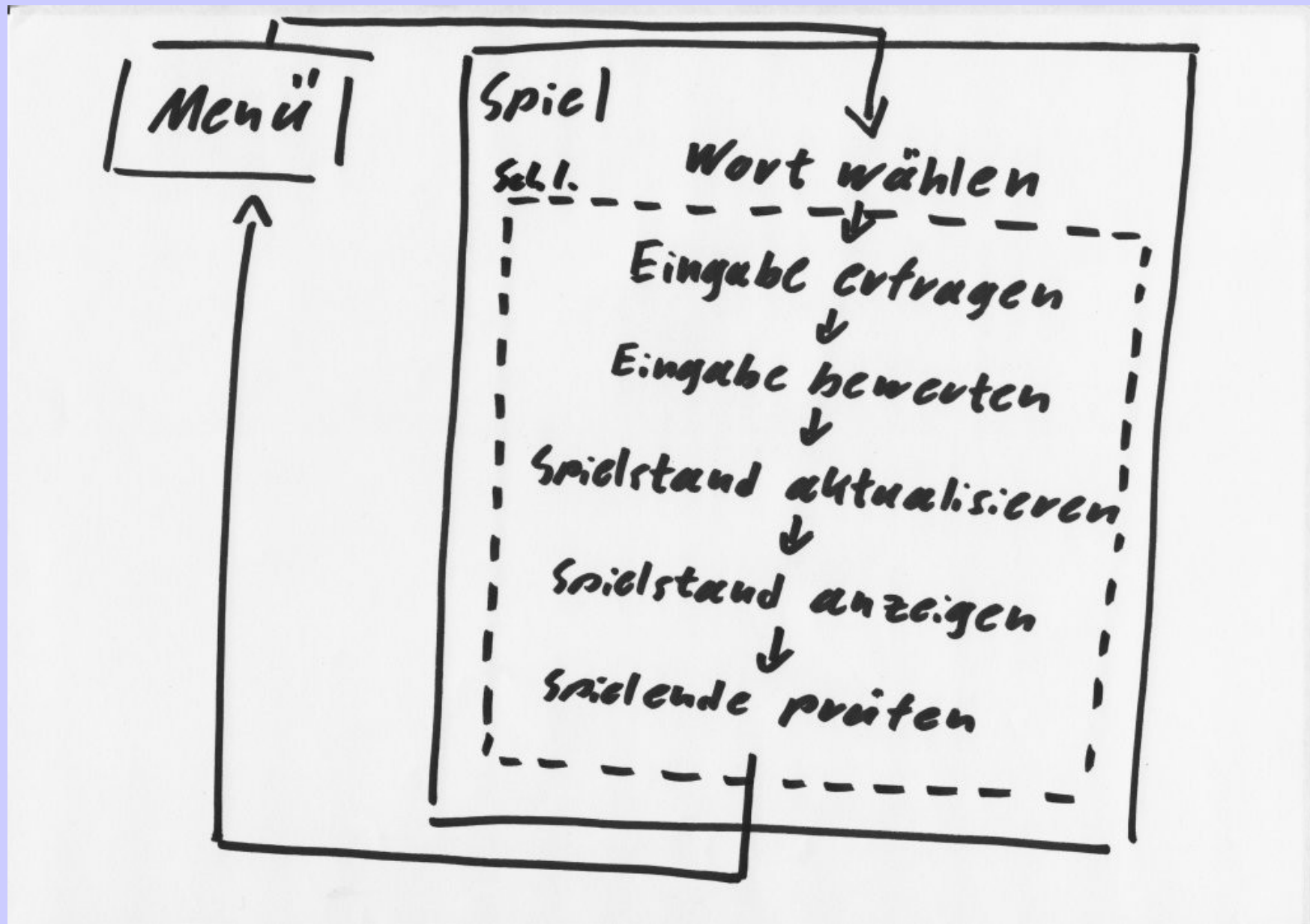
3. Ablauf ordentlich aufschreiben

- wenn der Platz in der Skizze für die Korrekturen nicht mehr ausreicht
 - Neues Diagramm
 - Erfahrungen aus der Skizze umsetzen
 - Aufgabe in möglichst viele Teilprobleme zerlegen

Beispiel eines Ablaufdiagramms



3.1 Ablauf nochmal durchdenken

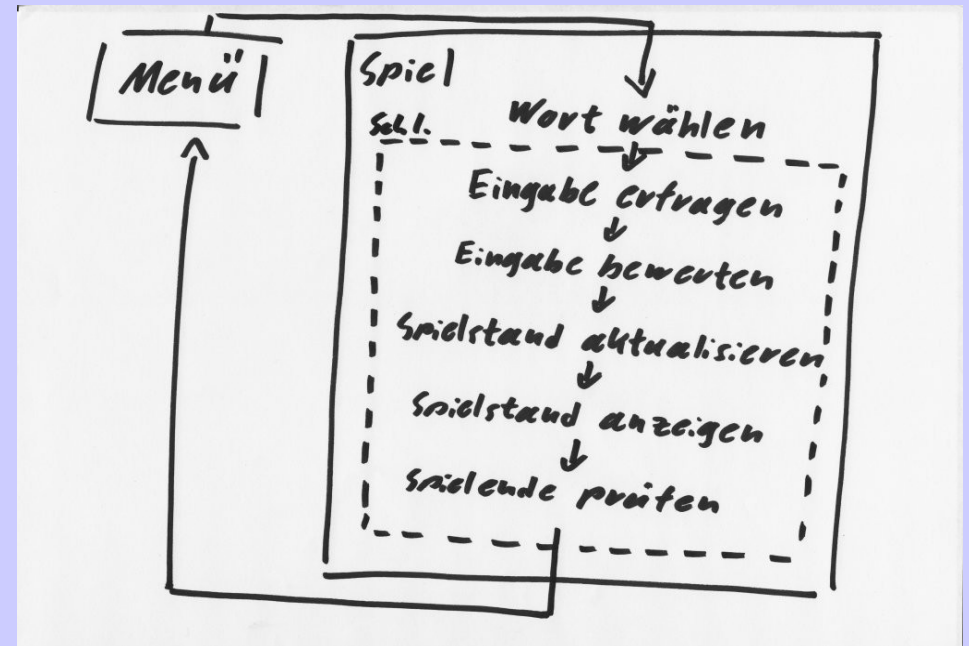


4. Methoden extrahieren

- Jedes Teilproblem stellt eine Methode da
 - sinnvoll benennen (sprechende Namen)
 - Übergabeparameter festlegen
 - was braucht die Methode zum Arbeiten ?
 - Rückgabeparameter festlegen
 - was liefert die Methode als Ergebnis ?

Beispiel für extrahierte Methoden

- ~~boolean menu ()~~
- boolean isGameWanted()
- String chooseWord ()
- char askForCharacter ()
- boolean isCharacterInWord (char character, String word)
- void updateGame (char guessedCharacter)
- void displayGame ()
- boolean isGameOver ()



5. Teilprobleme zerlegen

- Schritt 1 bis 5 für jedes Teilproblem wiederholen
 - erst aufhören wenn Teilprobleme trivial sind
- Am Ende ist die Gesamtlösung klar, wir müssen nur noch den Code schreiben

Wie behalte ich den Überblick?

Code schreiben

- Debugging is twice as hard as writing the code in the first place.
- Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.

Brian Kernighan

Ablauf durch Kommentare andeuten

```
public static void main(String args[]) {  
    // -Menueschleife-  
        // neues Spiel ? wenn nein, dann Ende  
        // Wort wählen  
        // -Spielschleife-  
            // Spielstand ausgeben  
            // Buchstaben erfragen  
            // Eingabe bewerten  
            // Spielstand aktualisieren  
            // Spielende prüfen  
        // -ende-  
    // -ende-  
}
```

Schrittweise Code erzeugen

- Niemand schreibt korrekten Code !
- Code muss getestet werden
 - so früh wie möglich
 - so oft wie möglich
 - so gründlich wie möglich
- Mit Teilproblemen beginnen die sich leicht testen lassen

Testbarkeit herstellen

- Durch Testausgaben

```
// TODO: delete testoutput  
System.out.println(„Zählerstand: “+counter);
```

- Durch Dummymethoden

```
public static char readChar(){  
    // TODO: implement readChar  
    return 'j';  
}
```

Flexibel sein

- Kein Plan überlebt den ersten Feindkontakt
- Probleme sind meist komplizierter als man denkt
 - Entwurf kann stetig angepasst werden
- Fehler in Lösungen entdeckt man wenn man sie präzise niederschreibt

Code lesbar halten

- Kapazität Kurzzeitgedächtnis: 3 – 7 Items
 - zwischen 3 und 7 Codezeilen können maximal gleichzeitig überblickt werden
 - größere Codeblöcke übersichtlich strukturieren
 - besser: in Methoden zerlegen
 - Fehlersuche schwer -> Code besser strukturieren

Verschachtelte Blöcke

```
if (isShip(row, column)) {  
    int ship = getShip(row, column);  
    if (getShipStatus(ship) == 0) {  
        if (getFleetStatus() == 0) {  
            gameLost = true;  
            this.notify();  
            return "Lost";  
        }  
        return „Sunk“;  
    }  
    return „Hit“;  
}else{  
    this.notify();  
    return „Water“;  
}
```

Ein wenig Hilfe fürs Gehirn

```
if (isShip(row, column)) {  
    // Schiff wurde getroffen  
    int ship = getShip(row, column);  
    if (getShipStatus(ship) == 0) {  
        // Schiff wurde zerstört  
        if (getFleetStatus() == 0) {  
            // Flotte wurde zerstört  
            gameLost = true;  
            this.notify();  
            return „Lost“;  
        }  
        return „Sunk“;  
    }  
    return „Hit“;  
}else{  
    // Schiff wurde verfehlt  
    this.notify();  
    return „Water“;  
}
```

Lesbar

```
// Schuss auf Spielfeld bei row/column  
if (isShip(row, column)) {  
    return processHit(row, column);  
} else {  
    return processMiss();  
}
```

An unserem Beispiel

```
public static void main(String args[]) {  
    // -Menueschleife-  
        // neues Spiel ? wenn nein, dann Ende  
        // Wort wählen  
        // -Spielschleife-  
            // Spielstand ausgeben  
            // Buchstaben erfragen  
            // Eingabe bewerten  
            // Spielstand aktualisieren  
            // Spielende prüfen  
        // -ende-  
    // -ende-
```

An unserem Beispiel

```
public static void main(String args[]) {  
    // -Menueschleife-  
        // neues Spiel ? wenn nein, dann Ende  
        // Wort wählen  
        // -Spielschleife-  
            // Spielrunde durchführen  
        // -ende-  
//-ende-
```

Ein Wort zur Performance

- First make it run, then make it run fast.

Brian Kernighan

- Compiler optimiert Performance des Codes
- Erst über Performance nachdenken wenn das Programm zu langsam ist

Zusammenfassung

- Analysephase
 - Überblick verschaffen
 - Teilprobleme identifizieren
 - Ordnung in das Chaos bringen
- Entwurfsphase
 - Ablaufplan erstellen
 - Teilprobleme zerlegen
 - Methoden extrahieren
- Implementierungsphase

