Kapitel 11.3: Klassen – Tetris

Weiter gehts mit Tetris! Zuerst wollen wir unseren bisherigen Code, der ja noch ziemlich unübersichtlich in einer einzigen Datei steht, aufräumen. Das machen gute Programmierer tatsächlich sehr häufig! Wir teilen unseren Code dazu auf, d. h. wir **modularisieren** ihn. Sinn einer Modularisierung ist, eine bessere Strukturierung zu erreichen und damit das Erweitern des Codes zu erleichtern. Dies alles erreichen wir, indem wir **Klassen** einführen!

Klasse Block

- Lege die Klasse `Block` in der Datei `block.py` an.
- Jeder Block hat die Attribute:
  - `x` # Aktueller x-Wert des Blocks (top-left corner).
  - `y` # Aktueller y-Wert des Blocks (top-left corner).
  - `size` # Seitengröße des Blocks.
  - `color` # Die Farbe, in der der Block gezeichnet wird.
  - `max_x` # Die Fensterbreite (weiter darf der Block nicht nach rechts).
  - `max_y` # Die Fensterhöhe (quasi der Boden).
  - `speed` # Die Geschwindigkeit in Pixeln, mit der der Block fällt.
  - `active` # `True`, falls der Block noch nicht unten bei max_y angekommen ist, sonst `False`
- Die Klasse Block hat die folgenden Funktionen:
  - `__init__` → Initialisiere die Attribute des Blocks - welche Parameter brauchst du dazu?
  - `draw` → Der Block wird im Fenster gezeichnet.
  - `is_active` → Liefert den Wert des Attributs active zurück.
  - `drop` → Wenn der Block noch aktiv ist, "fällt" er um speed pixel nach unten
- Benutze jetzt Objekte der Klasse Block in deinem Hauptteil

Klasse Board

- Lege eine Klasse Board an (auch unser Spielfeld ist ein Objekt und kann aus der `main.py` ausgelagert werden)
- Das Board hat folgende Attribute:
  - `block_size`
  - `blocks_per_row`
  - `blocks_per_col`
  - `width`
  - `height`
  - `active_block`
- Das Board hat folgende Methoden:
  - `__init__`
  - `draw`
  - `play`
  - `get_width`
  - `get_height`

- In der Klasse `main` gibt es jetzt nur noch die Gameloop, in der ein Board erzeugt wird. Hier wird in jedem Durchlauf `board.play` und `board.draw` aufgerufen.

Blöcke am Boden sammeln

Erweitere jetzt dein Tetris-Projekt schrittweise so, dass Blöcke, die auf dem Boden ankommen, in einer List gespeichert werden.

- Erweitere die Klasse Board um das Attribut `inactive_blocks`. Dieses Attribut ist eine Liste, in der alle Blöcke gesammelt werden, die auf dem Boden angekommen, also nicht mehr aktiv sind. Zu Beginn ist diese Liste leer.
- Wenn der aktive Block also inaktiv wird, wird er der Liste `inactive_blocks` hinzugefügt und ein neuer aktiver Block wird erzeugt.
- Auch die inaktiven Blöcke müssen gezeichnet werden!

Blöcke in Intervallen erzeugen

Wenn du genau hinsiehst, dann stellst du fest, dass die Blöcke nicht an jeder beliebigen x-Position sein können. Wenn wir später Tetris spielen, wäre es aber besser, wenn die Blöcke nur in Abständen von `block_size` anfangen könnten. In unserem Fall also bei `x = 0`, `x = 20`, `x = 40`, ..., `x = 300`. Erweitere dein Programm so, dass die Blöcke nur noch an x-Positionen erzeugt werden, die durch `block_size` ohne Rest teilbar sind.