Model vs. View & HangMan

KSR

EF Informatik

Model vs. View

- Programmierparadigma: «Programmier-Philosophie»
- Idee: **Strikte Trennung** von Code, der für das **Model(l)** verantwortlich ist und von Code, der für die **Anzeige (View)** verantwortlich ist.
- Vorteil 1: Code sauber strukturiert, einfacher verständlich
- Vorteil 2: Code kann einfach ‘recycled’ werden, verwende gleichen Model-Code für verschiedene Views.

Beispiel: Asteroid Game

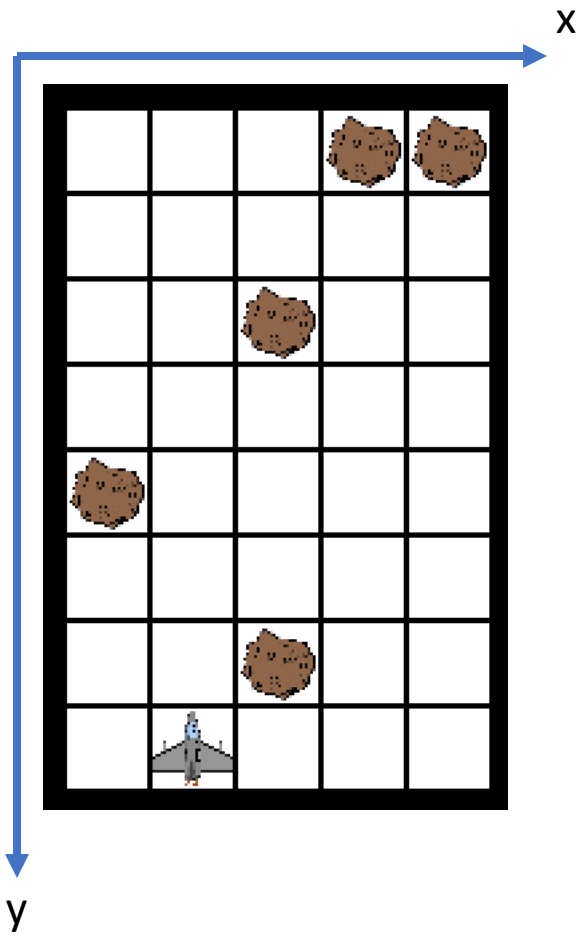
- Model:
  - Aktueller **Game State** muss **modelliert**, also im **Code abgebildet** werden.
  - Möglichkeiten für Spaceship und Asteroiden?

```
// SPACESHIP
// Variante 1
int spaceshipX = 1;
int spaceshipY = 7;

// Variante 2
int[] spaceship = new int[] { 1, 7 };
```

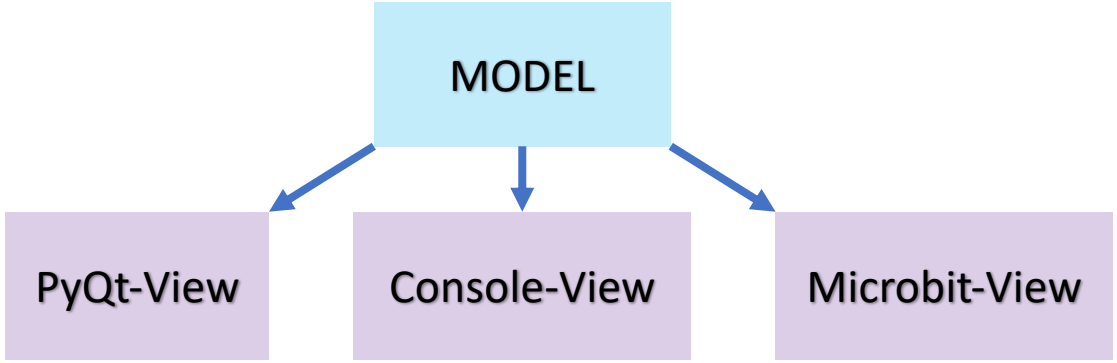
```
// ASTEROIDS
// Variante 1
int[,] asteroidsGrid = {
    { 0, 0, 0, 1, 1 },
    { 0, 0, 0, 0, 0 },
    { 0, 0, 1, 0, 0 },
    { 0, 0, 0, 0, 0 },
    { 1, 0, 0, 0, 0 },
    { 0, 0, 0, 0, 0 },
    { 0, 0, 1, 0, 0 },
    { 0, 0, 0, 0, 0 }
};

// Variante 2
int[,] asteroidsList =
{
    {3, 0}, {4, 0}, {2, 2}, {0, 4}, {2, 6}
};
```

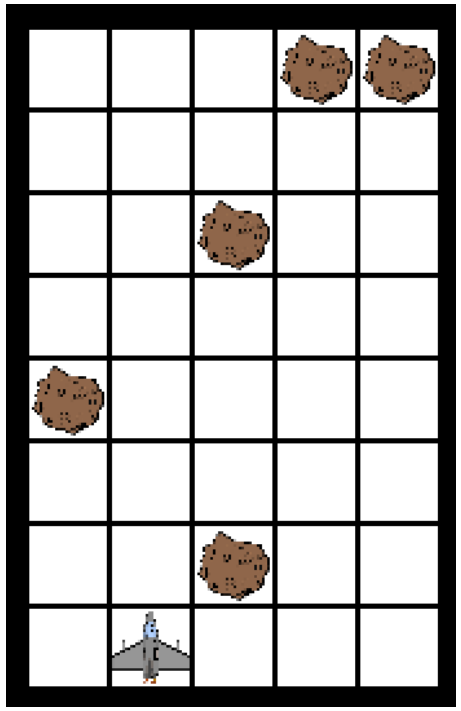


- View:
  - Zuständig, Variablen des Modells richtig zu interpretieren und auf Bildschirm anzeigen

Beispiel: Asteroid Game

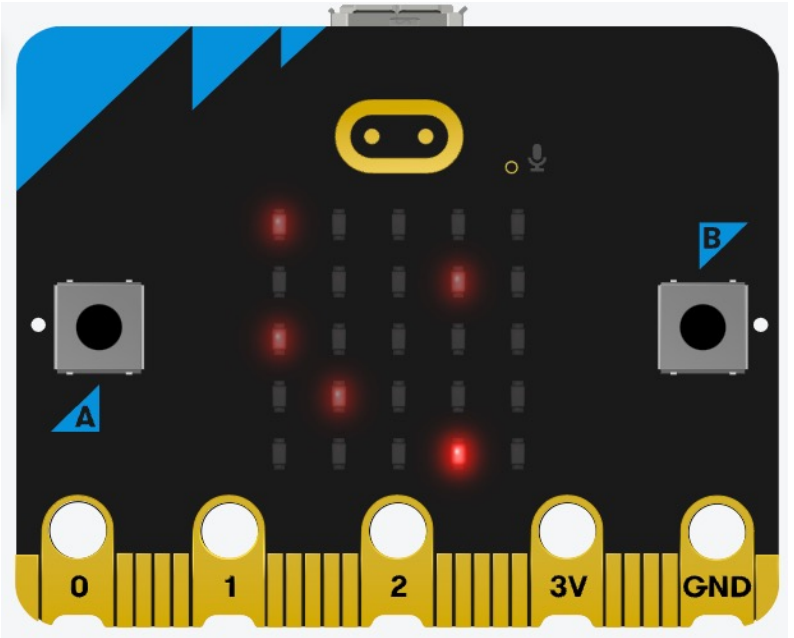


- Python-Projekt aus TALIT:
  - 1x Modell programmiert
  - 3x verschiedene Views programmiert, die alle das identische Modell verwenden
  - Weitere Ideen: Web-Game, Mobile-Game, …



PyQt (Bibliothek für GUI

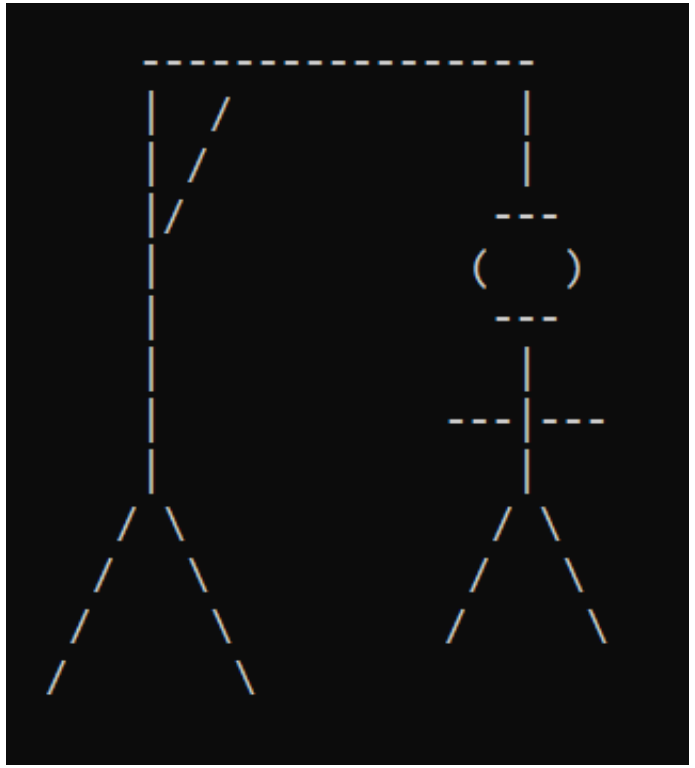
Konsolenapp



Microbit

Projekt: HangMan

- **Ziele:**
  - Programmiere das bekannte **HangMan**-Game als **Konsolenapp**.
  - **Grösseres Programm** schreiben (ca. 300 Zeilen Code)
  - Lerne, wie man anhand des Programmierparadigmas **Model vs. View** Code sinnvoll strukturiert.
  - **Scope** von Variablen verstehen.
  - Unterschied **Werttypen vs. Referenztypen** verstehen.
  - **Robusten Code** schreiben, der für alle Eventualitäten gewappnet ist.
- **Demo!**



Template

Model-Code

View-Code

```
namespace HangMan; // here use name of your project

class Program
{
    // No variable declarations in this area!!
    static void Main(string[] args)
    {
        // Variable declarations allowed here
        while (true)                    // The game repeats until finished by player 1
        {
            ReadSecretWord();           // Player 1: Enter the secret word to be guessed by player 2
            HangTheMan();               // Screen output for a good start
            while (true)                // Player 2: Make your guesses
            {
                ReadOneChar();          // Handle input of one char.
                EvaluateTheSituation(); // Game Logic goes here
                HangTheMan();           // Screen output goes here
            }
            QuitOrRestart(); // Ask if want to quit or start new game
        }
    }

    static void ReadSecretWord() // Modification of method declaration recommended: Add return value and parameters
                                 // If there are rules and constraints on allowed secrets (e.g. no digits), check them in here
    {
        // Variable declarations allowed here
        // Console.Write() etc. allowed here!
    }
```

Verschiedene Aufgaben in **Funktionen ausgelagert** -> **Main-Methode schlank** und übersichtlich

Willst **andere Version** von HangMan? Z.B. GUI-, Web-, Mobile Game?

➔ Muss **nur 4 View-Funktionen neu programmieren**. Alles vom Model kann 1:1 übernommen werden!

Projekt: HangMan

- **Zeit:** 2x3Lektionen
- Kann auch weniger makabere graphische Darstellung wählen
- **Auftrag auf Wiki** (studiere diesen genau!)