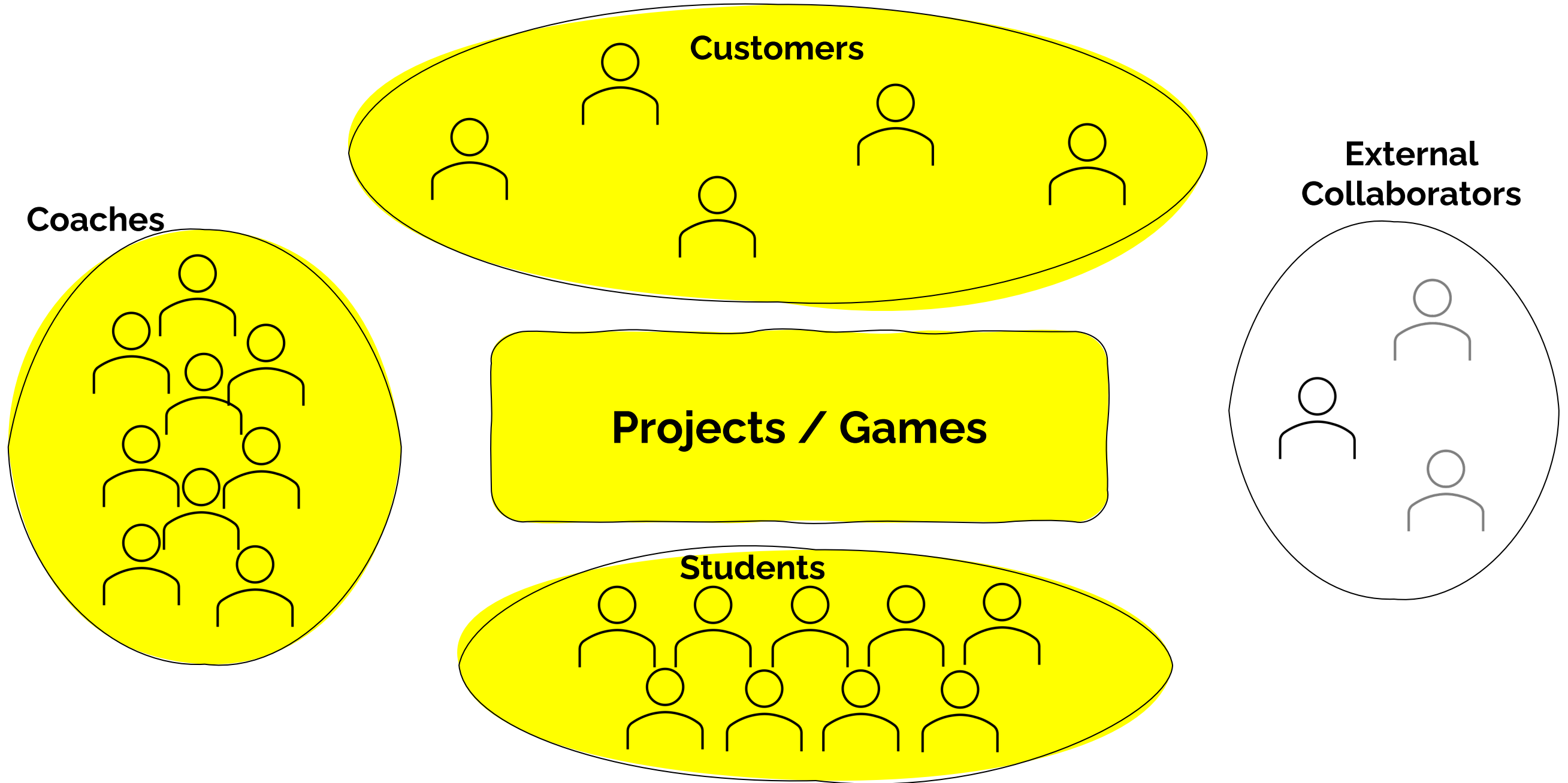


ConnectFHNW

Games from students for students

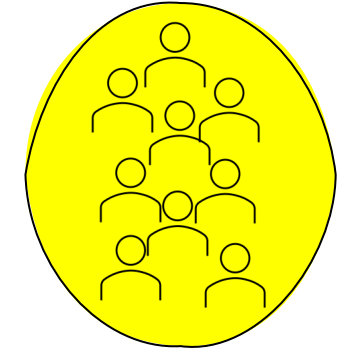




Prof. Dr. Barbara Scheuner

- Co-Head of the BCs Programm in Computer Science
- Background: Major in Theoretical Computer Science and Minor in Didactics from ETH
- I love to play games, work with my hands and teach

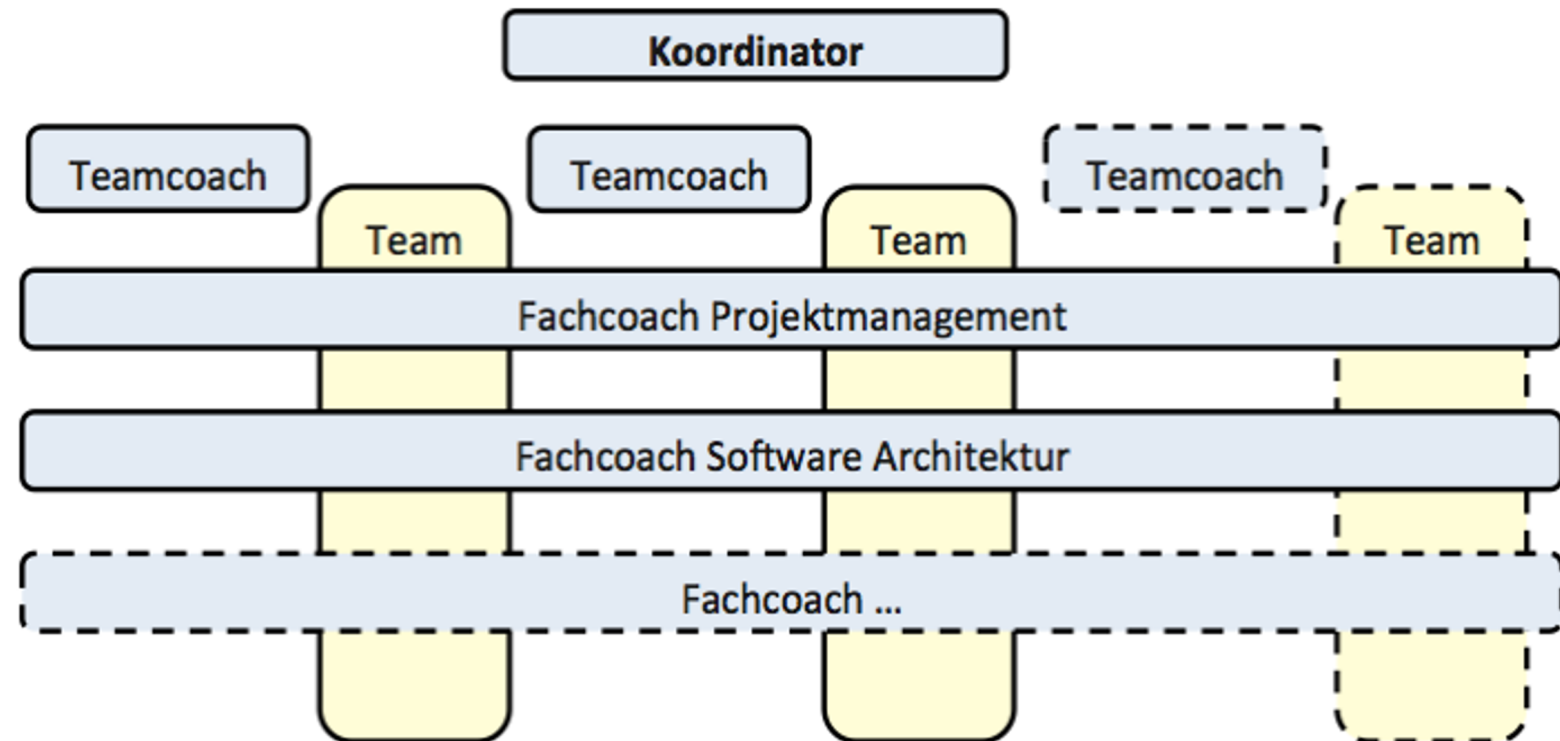




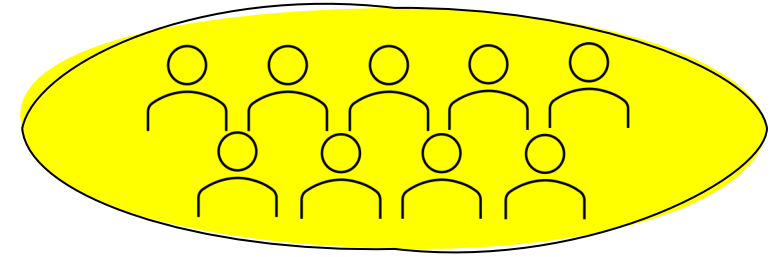
Coach Team

Different Coaches:

- Project Management
- Requirements Engineering
- Usability
- Information Management
- Software Engineering
- Source Code
- Testing



Each team also has a team coach for organizational questions.



Students (computer science students)

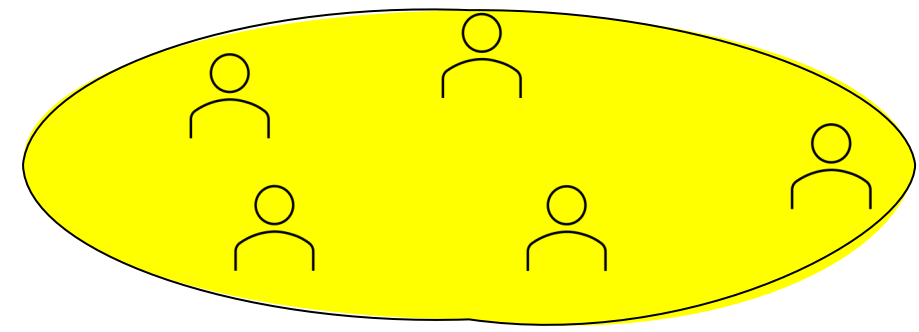
Groups of 7-9 students with different experiences and interests.

The student group (55 Students) of this year consists of:

- 15 Women / 40 Men
- 30 Computer Science / 25 Profile iCompetence

Backgrounds:

- 16 Computer Science apprenticeship
- 12 KV
- Others like: Automation technician, clothing designer, electronics technician, information and documentation specialist, building services planner, geomatics technician, chemistry laboratory technician, media technician, primary school teacher, production mechanic, architectural drawer

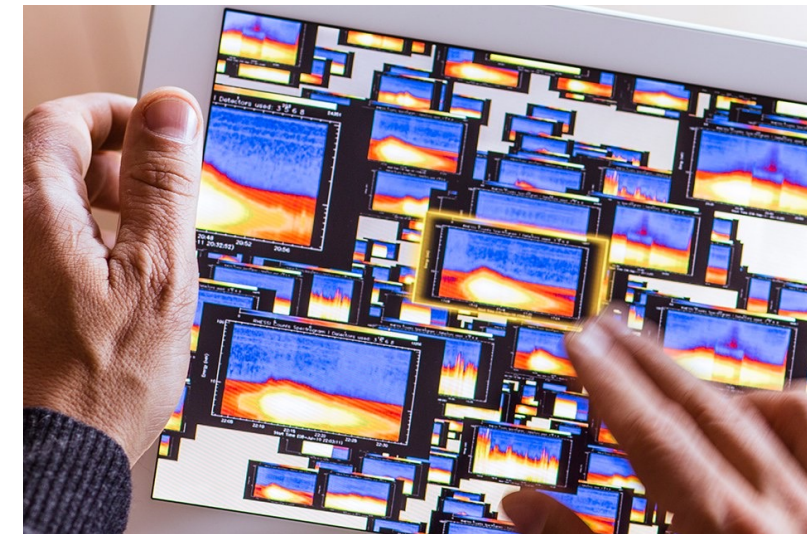


Customers

The customers are employees of the FHNW School of Engineering.

They are either

- Teachers, who are teaching courses for the CS-Students. Currently, mostly mathematical courses.
- Scientific staff from an institute (IIT, IMVS, I4DS)

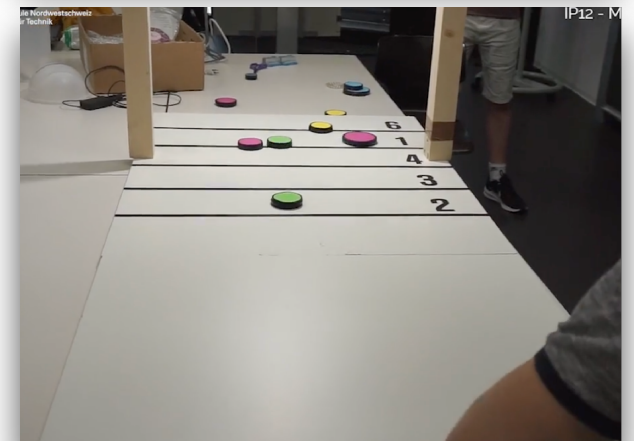
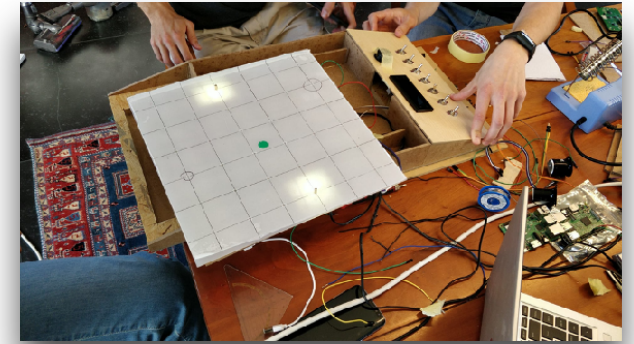


Projects

All projects use a RaspberryPi and Java to build a computer science related game.

The games typically are:

- Board Games
- Arcade Games
- Natural Games



100%
Pure
Java



F X G L

Board Game

Board games use a highly sophisticated board with multiple sensors and output devices such as:

- Buttons
- RFID Readers
- LEDs (also available as Matrix)
- Speakers

Technology used:

- Java 17 + Pi4J V2



Arcade Game

Arcade games have a display and a controller. They can be played on a «normal» computer.

Technology used:

- FXGL
- Java 17 + Pi4J V2



Natural Game

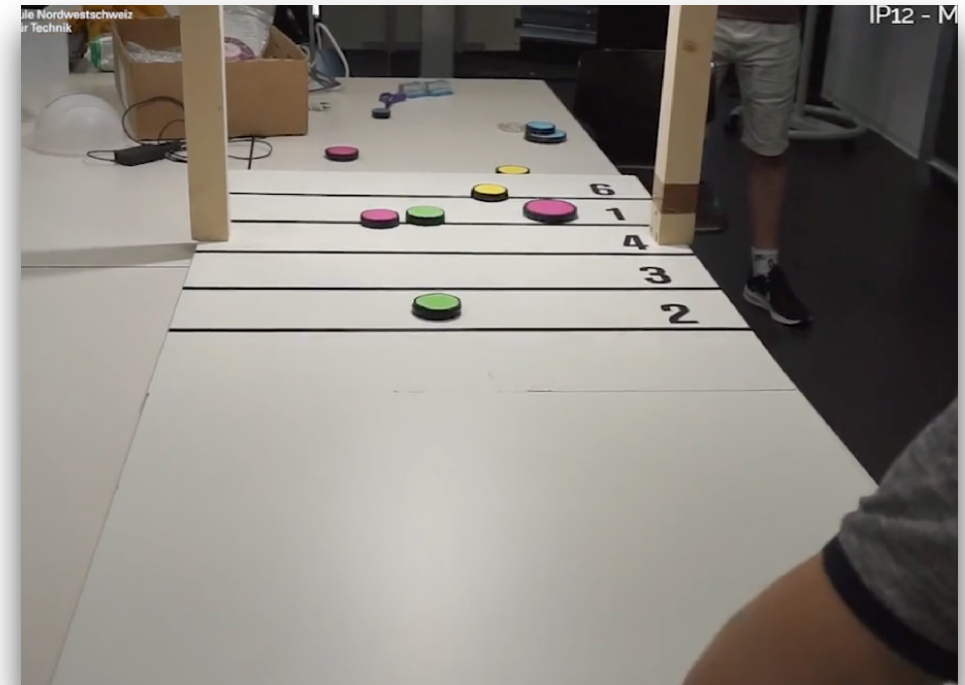
Natural games consist of physical parts which are recognized via a camera.

They may include a controller to support the game.

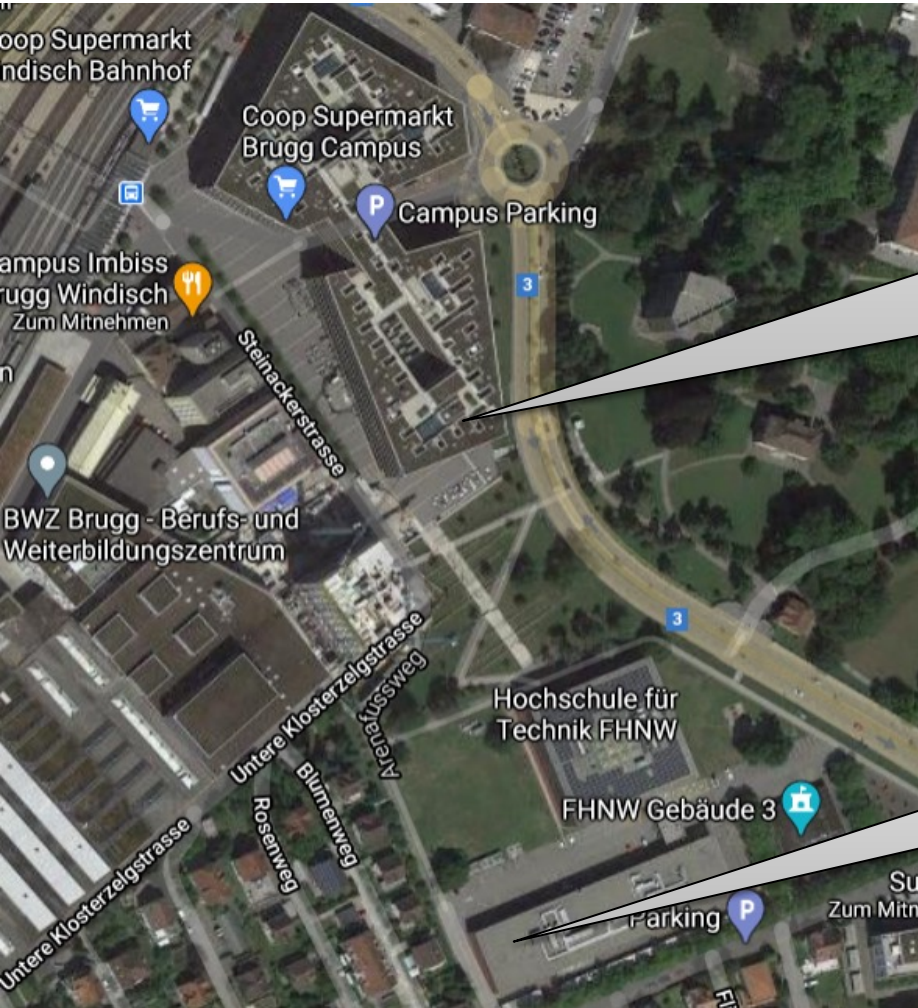
The status of the game is calculated on the basis of snapshots.

Technology used:

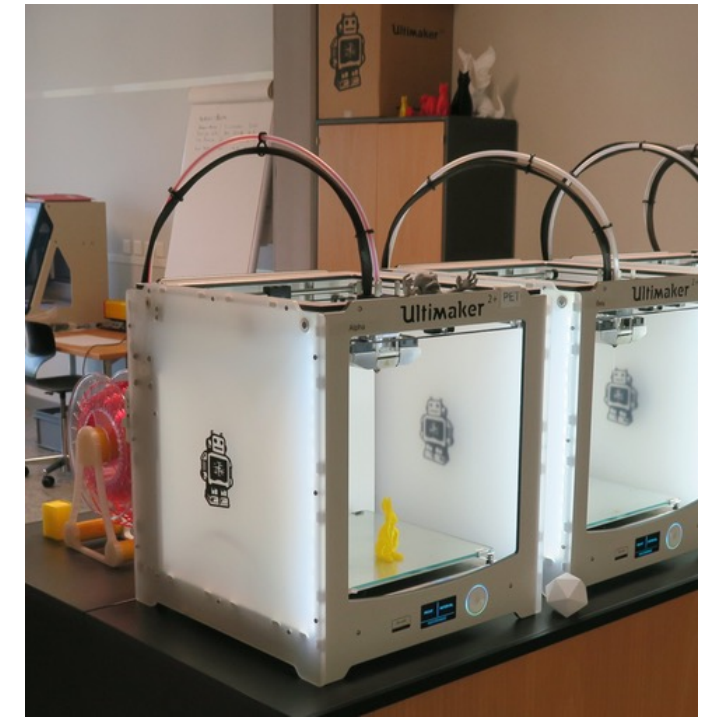
- JavaCV
- Java 17 + Pi4J V2



Maker facilities



Maker Studio with 3D printers.
<https://web.fhnw.ch/plattformen/makerstudio/>



Inhouse factory for laser cutting

Structure and learning environment

We give an overall timetable with some fixed dates.

Kickoff-Week, team building, contact with first Pi4J project

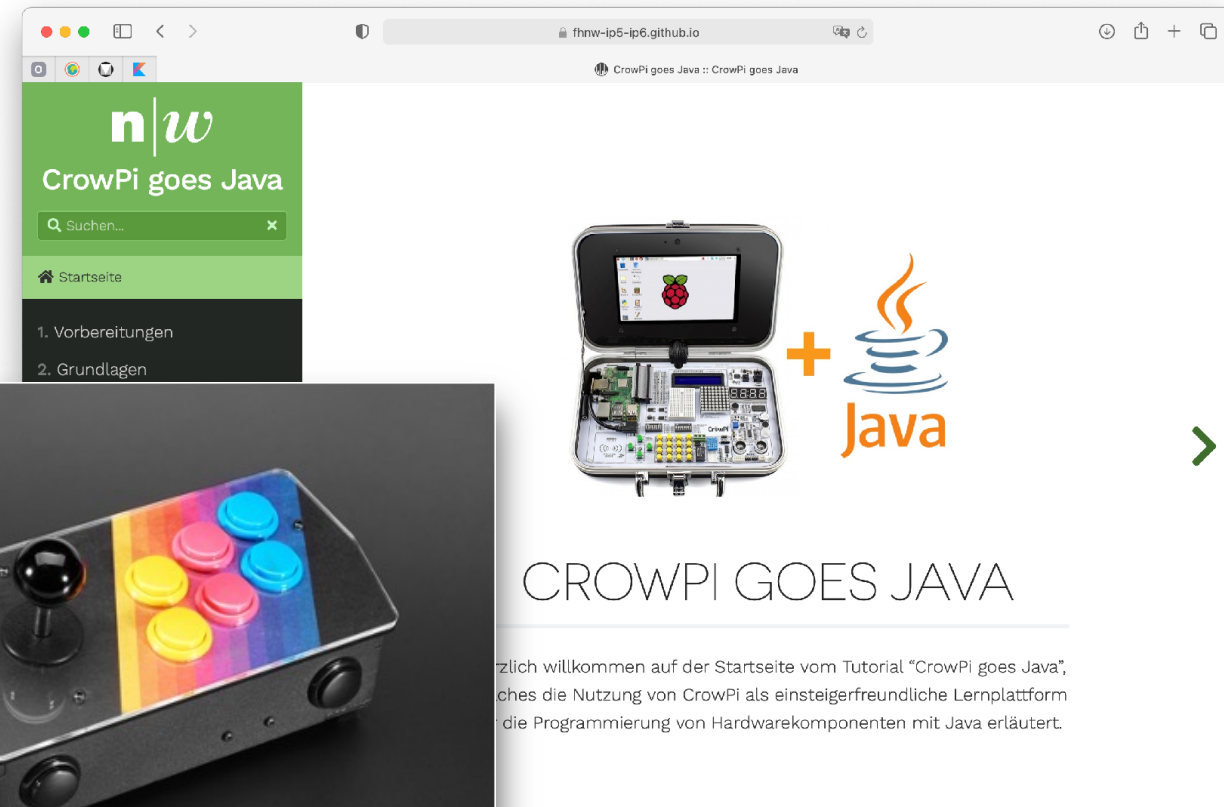
Project week: one week without additional courses

	Herbstsemester													Frühlingssemester																															
Jahr	2021													2022																															
Kalenderwoche	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24					
Montag - Samstag	13.09. - 18.09.	20.09. - 25.09.	27.09. - 02.10.	04.10. - 09.10.	11.10. - 16.10.	18.10. - 23.10.	25.10. - 30.10.	01.11. - 06.11.	08.11. - 13.11.	15.11. - 20.11.	22.11. - 27.11.	29.11. - 04.12.	06.12. - 11.12.	13.12. - 18.12.	20.12. - 25.12.	27.12. - 01.01.	03.01. - 08.01.	10.01. - 15.01.	17.01. - 22.01.	24.01. - 29.01.	31.01. - 05.02.	07.02. - 12.02.	14.02. - 19.02.	21.02. - 26.02.	28.02. - 05.03.	07.03. - 12.03.	14.03. - 19.03.	21.03. - 26.03.	28.03. - 02.04.	04.04. - 09.04.	11.04. - 16.04.	18.04. - 23.04.	25.04. - 30.04.	02.05. - 07.05.	09.05. - 14.05.	16.05. - 21.05.	23.05. - 28.05.	30.05. - 04.06.	06.06. - 11.06.	13.06. - 18.06.					
Herbstsemester		1	2	3	4	5	6	7	8	9	10		11	12	13		14	15																											
Frühlingssemester																									1	2	3	4	5	6	7	8			9	10			11	12	13	14	15		
Einführungswoche für 1. Sem.	E																																												
Projektwoche												P																																	

Technological support

We provide some «playgrounds» with ready to start projects:

- CrowPi (<https://fhnw-ip5-ip6.github.io/CrowPiGoesJavaTutorial/de/>)
- Picade
- FXGL-Tutorials



n|w
CrowPi goes Java

Suchen... x

Startseite

1. Vorbereitungen

2. Grundlagen

Raspberry Pi + Java

CROWPI GOES JAVA

zlich willkommen auf der Startseite vom Tutorial "CrowPi goes Java",
ches die Nutzung von CrowPi als einsteigerfreundliche Lernplattform
die Programmierung von Hardwarekomponenten mit Java erläutert.

Ready made OS Images

Pi4J-OS

- pure Pi4J-projects, pure JavaFX-projects, pure FXGL-projects
- integrated JavaFX/Pi4J projects

CrowPi-OS

- for all experiments on CrowPi

Picade-OS

- FXGL-projects for Picade

GameHAT-OS

- FXGL-projects for GameHAT

Game Design



3 project ideas

1 final idea

Board Game



Pi4J Tutorial



selected components

Arcade Game



Pi4J Tutorial

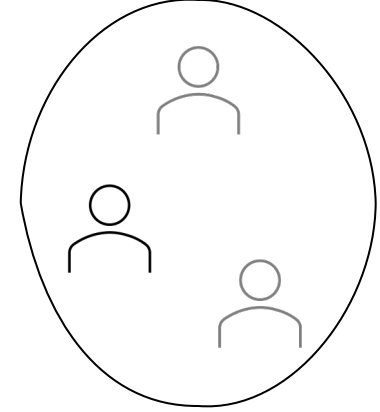


FXGL Tutorial



selected components

Building of the game



External Collaborators

To connect our students to open source collaborators and to share our knowledge we participate in the Pi4J (<https://pi4j.com/about/>) project.

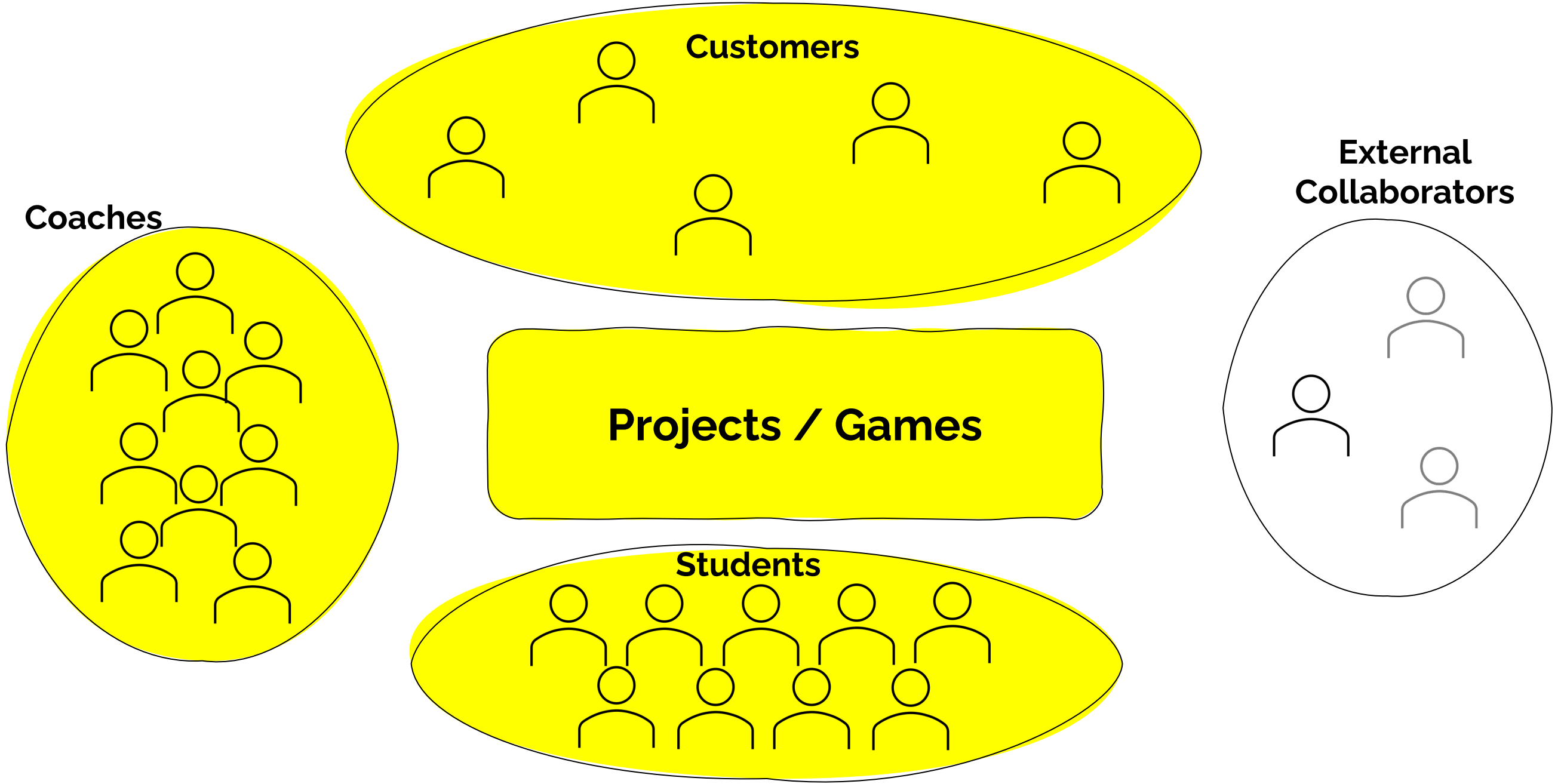
- Frank delPorte from Belgium (<https://github.com/FDelporte>)
- Robert von Burg (<https://github.com/eitch>)

For the arcade games we are using FXGL:

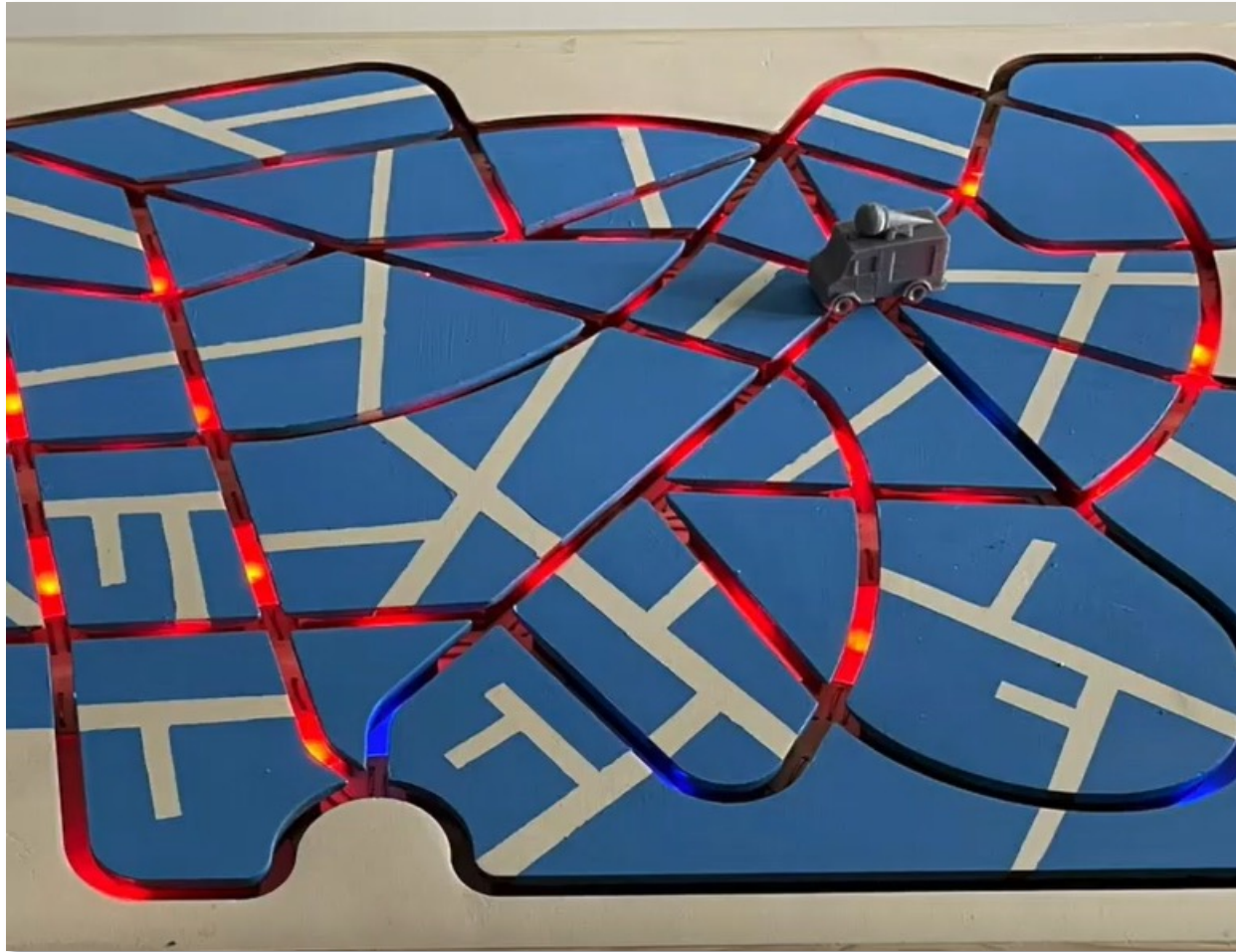
- Almas Baimagambetov from the University of Brighton: <https://github.com/AlmasB>

For Support concerning JavaFX on the RaspberryPi:

- Gluon: <https://gluonhq.com/>



Videos



Finally:

**Final Event:
15.6.2022
@FHNW:Windisch**

If you want to see and probably play this year's games just let me know:

barbara.scheuner@fhnw.ch

Libraries

Java 17

- <https://adoptium.net>

Pi4J V2

- <https://pi4j.com>

JUnit

- <https://junit.org/junit5/>
- auch für Pi4J - Komponenten

FXGL

- <http://almasb.github.io/FXGL/>

JavaFX 17

- <https://openjfx.io>

JavaCV

- <https://github.com/bytedeco/javacv>

MQTT Client

- <https://github.com/hivemq/hivemq-mqtt-client>



Template Project

DRM MVC (vintage)
Native APP Generation (via GraalVM)
(Direct Rendering Mode)

through the application.

The MVC concept

The classic Model-View-Controller concept contains in addition to the starter class at least 3 more classes. The interaction is clearly defined:

```

    graph TD
      GUI[GUI] -- trigger actions --> Controller[Controller]
      Controller -- trigger actions --> PUI[PUI]
      Controller -- update state --> Model[Model]
      Model -- visualize state --> GUI
      Model -- visualize state --> PUI
  
```

The diagram illustrates the MVC pattern with three main components: GUI, Controller, and Model. The GUI (labeled 'GUI of a Pi4J App') shows a large number '73'. The Controller provides actions like 'list of methods' and 'e.g. increase, decrease'. The Model manages the whole state, including a 'list of ObservableValues' such as 'counter, on'. The PUI (Physical User Interface) is represented by a red LED and a black button. Arrows indicate the flow of data: GUI triggers actions on the Controller, which triggers actions on the PUI. The Controller updates the state of the Model, which then visualizes that state back to both the GUI and the PUI. Handwritten notes specify that the GUI is 'completely independent of PUI', the Model 'doesn't know anything about GUI and PUI', and the PUI is 'completely independent of GUI'.

- *Model classes*
 - contain the complete state which is to be visualized, thus these classes are called *Presentation-Model*