



FormalZ : Playful Formal Method

Wishnu Prasetya, Craig Leek, Orestis Melkonian, Jorris ten Tusscher
Universiteit Utrecht

IMPRESS Project

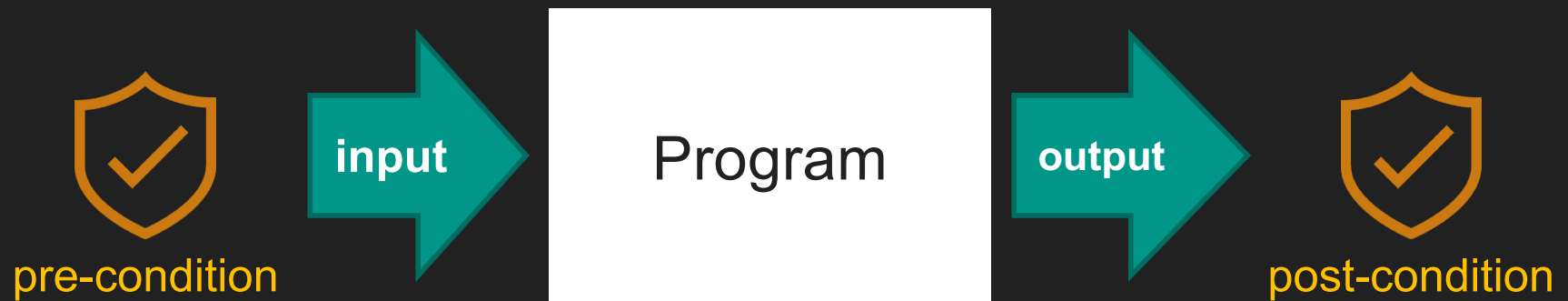
<https://impress-project.eu/>



ERASMUS+

Project 2017-1-NL01-KA203-035259

Why is it difficult to write bug-free software?



Unfortunately in practice people do not make these “guards” explicit enough.

Benefit of formal specifications

- One source of truth
- General, as opposed to concrete value test oracles
- Facilitate automated testing
- No, it does not require separate tooling --with λ -expression we can write specifications in-code.
- But yes... it does require some learning to appreciate it and become proficient with it.

Example of in-code specifications

```
public static void getMax_spec1(int[] a) {  
    // preconditions  
    pre(a != null);  
    pre(a.length > 0);  
  
    // call the actual function implementation  
    int retval = getMax(a);  
  
    // postconditions  
    post(exists(a, i -> a[i] == retval)); // A  
    post(forall(a, i -> a[i] <= retval)); // B  
}
```


A lesson in writing formal specifications

- A **formula** is either:
 - a simple formula
 - $\forall \text{identifier} \in \text{simple-expression} \bullet \text{formula}$
 - $\exists \text{identifier} \in \text{simple-expression} \bullet \text{formula}$
- For example:
 - $\forall x \in \text{Students} \bullet x.\text{age} \geq 16$
 - $\exists x \in \text{Students} \bullet x.\text{age} = 16$

A lesson in writing formal specifications



FormalZ : playfully formal

The screenshot displays the FormalZ game interface, which is designed to look like a retro computer game. The top status bar shows a coin icon with the value 4872, an hourglass icon, a heart icon, a trophy icon, and three buttons: SFX: Off, Music: Off, and Exit.

The main game area features a green circuit board map. In the center, a large blue-bordered box labeled "Precondition" contains a logic diagram. This diagram includes a "Begin" node, a comparison node "w == 0", and a "Post" node. Arrows indicate the flow of logic. Above the diagram are buttons for "Sell block", "Precondition", and "normalize". To the right of the diagram is a vertical list of logic blocks, each with a cost of 40 coins. The list includes a "previous" button, a "4/7" block, a "||" block, a "!" block, an "==" block, an "!=" block, and a "next" button. A red "X" icon is visible next to the "previous" button.

On the right side of the interface, there is a "Tower Info" section and a "Tower Shop". The "Tower Shop" lists five tower types with their respective costs:

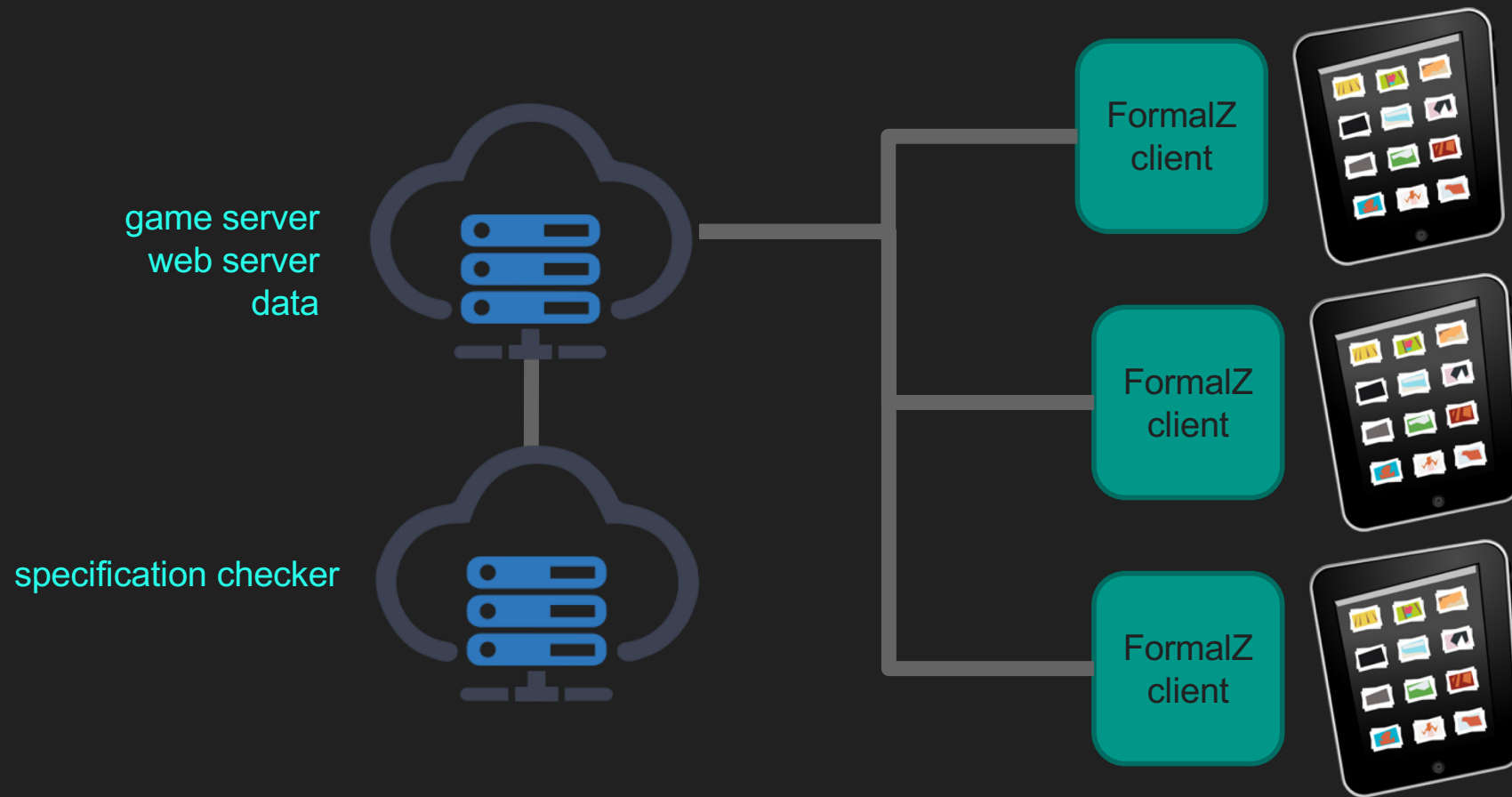
- Singleshot Tower (300 coins)
- Circle Tower (400 coins)
- Multishot Tower (400 coins)
- Pierce Tower (500 coins)
- Sniper Tower (1000 coins)

At the bottom of the interface, there are three blue buttons: "Desc", "Pre", and "Post". The "Pre" button is currently selected, and it displays the text "Pre: (w == 0)". To the right of these buttons are three more buttons: "Sell Tower", "Hints", and "Next Wave".

The game concept

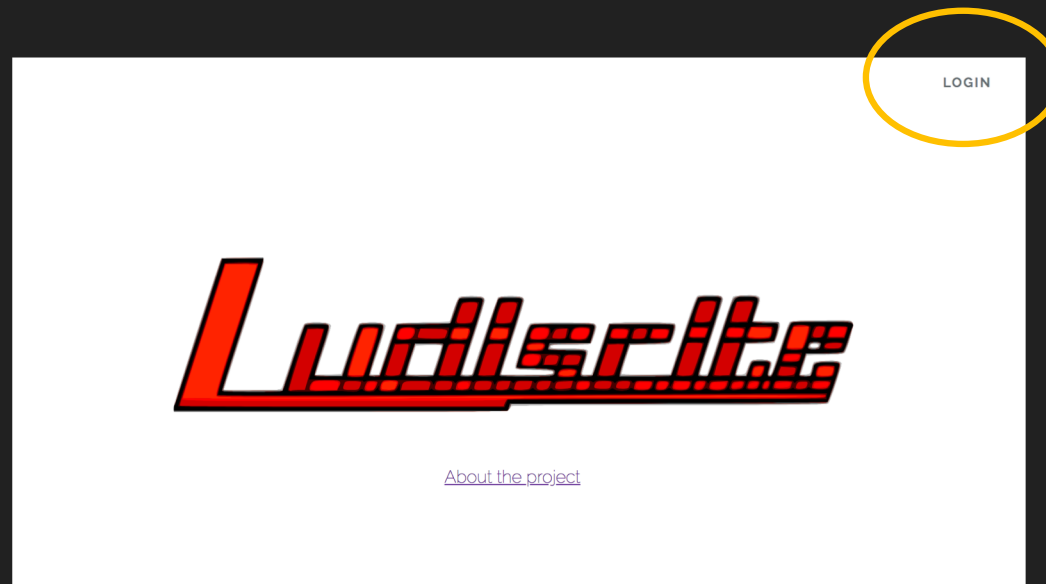
- It is an “educative game”, but not a trainer software.
- A cross-genre game of tower-defense and construction game
 - along the way, you also learn to formalize requirements.
- Deploy it in your course as a means to improve the students engagement.

FormalZ Architecture



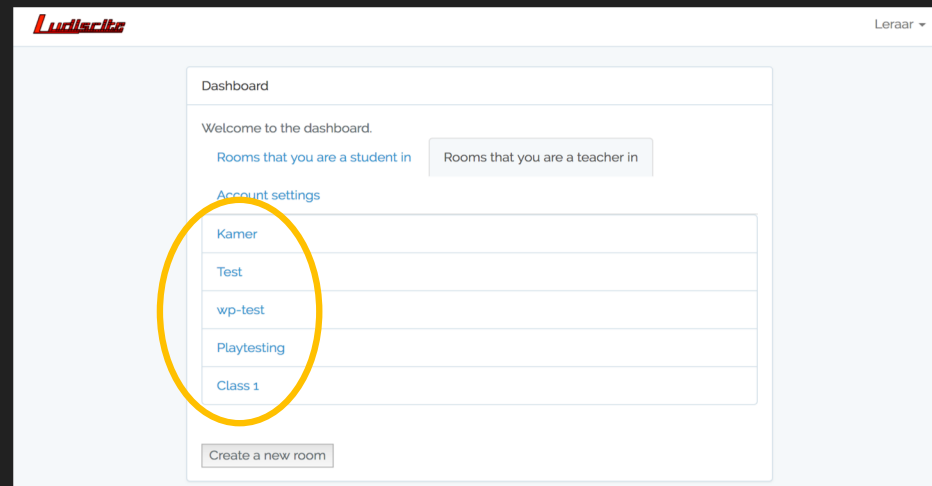
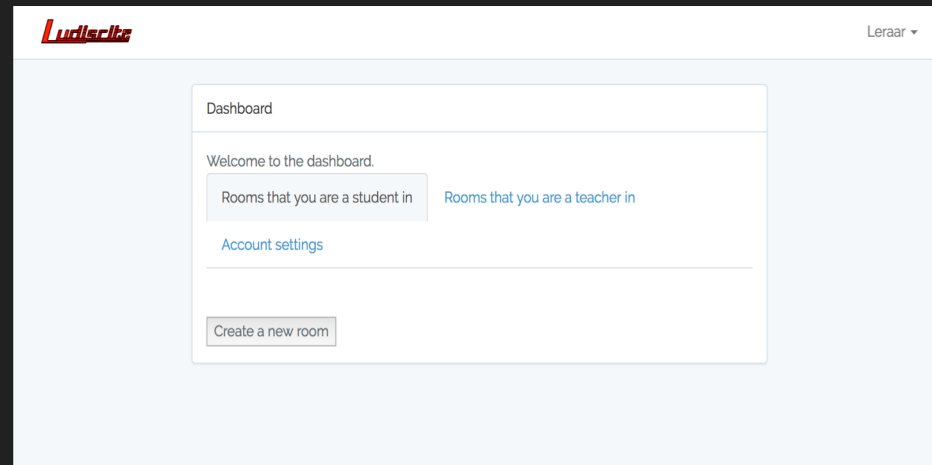
Tutorial FormalZ

<https://science-vs160.science.uu.nl/>



uname: funteacher
pwd: forall(h,a->h[a]<3)

Classroom management



Defining a “problem”

Edit problem

Problem Header	<input type="text" value="int add1p(int x)"/> <small>In the header, make sure to declare the types of EVERY variable in the problem!</small>
Problem Description	<input type="text" value="If x is a positive integer, this program re"/>
Preconditions	<input type="text" value="x > 0"/>
Postconditions	<input type="text" value="retval == (x + 1)"/>
Difficulty (1 - 5)	<input type="text" value="1"/> <input type="text" value="Examples for different difficulties"/>
Amount of intermediate problems (0 - 10)	<input type="text" value="1"/>

Few examples

- `int add1p(int x)`

If x is a positive integer, this program returns $x+1$. Use "retval" to denote the return value.

pre: `x > 0`

post: `retval == (x + 1)`

Few examples

- **boolean allzero(int[] a, int i, boolean retval)**

Given a non-null array a, the program allzero checks if a consists of only 0's. The return value (represented by retval) will indicate this.

pre: **a != null**

post: **retval == forall(a, i -> a[i] == 0)**

Classroom progress

```
int add1(int x)
```

If x is a positive integer, this program returns x+1. Use "retval" to denote the return value.

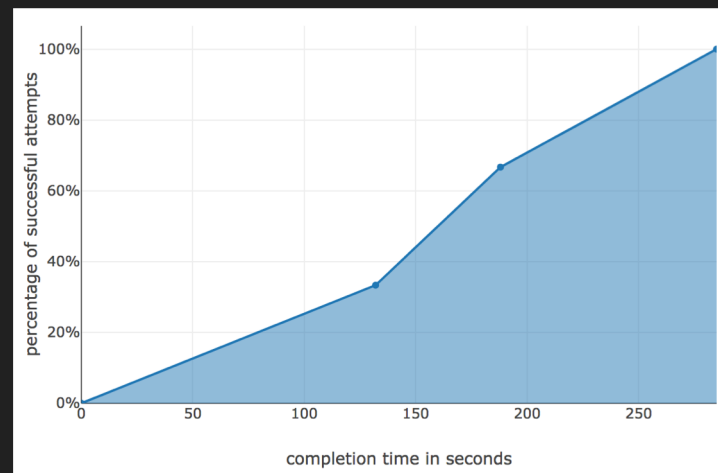
[Edit Problem](#) [Play Problem](#) [Back to Room](#)

[Highscores](#) [Your scores](#) [Your last game](#) [Statistics](#)

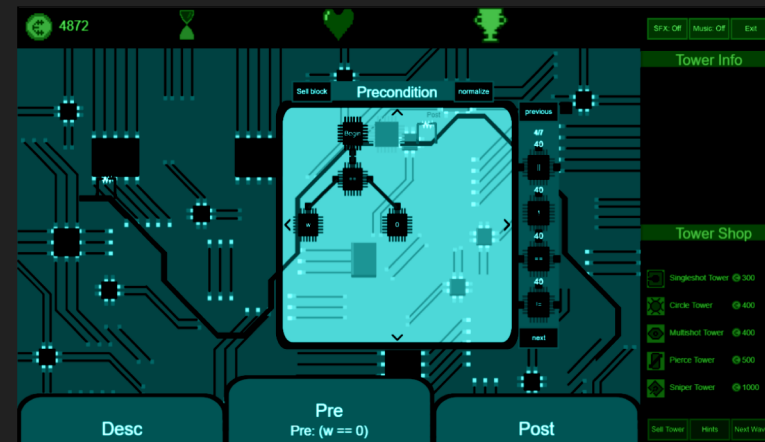
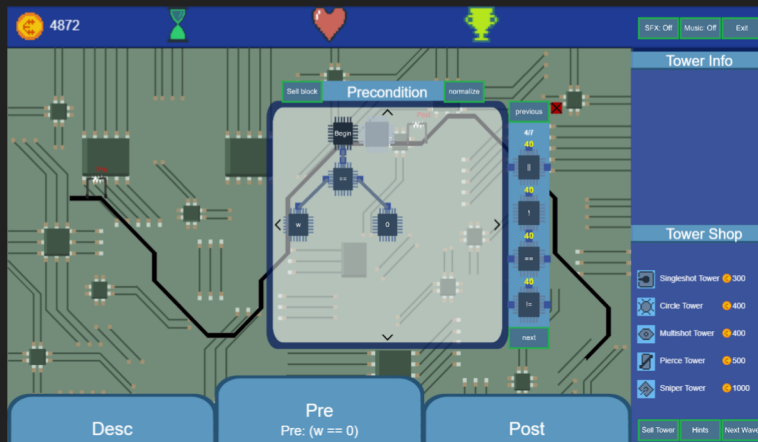
Student completion

Students who have completed the problem

#	Email
1	uprime812@gmail.com



Formalz future work



- Graphics.
- Analytics.
- Strengthening the gamification elements.
- Studying these innovations in actual class rooms.
- Public release.