EvaWeb: A Web App for Simulating the Evacuation of Buildings with a Grid Automaton*

André Greubel$^{1[0000-0002-7915-6520]}$, Hans-Stefan Siller$^{1[0000-0003-1597-7108]}$, and Martin Hennecke^{1}

University of Wuerzburg
andre.greubel@uni-wuerzburg.de

Abstract. In this Demo-Paper, we present EvaWeb, a Web Application for simulating the evacuation of buildings with a grid automaton. It is designed to support learning in the domain of mathematical modelling based on real-world problems. EvaWeb allows for 1) creating scenarios consisting of floor plans and persons within the environment, 2) loading and storing these scenarios to a text string, 3) automatically executing these scenarios, and 4) configuring the aesthetics and algorithms used during the exectuion. EvaWeb can be used online for free at https://evadid.it/eva2

Keywords: Simulation · Building Evacuation · Mathematical Modelling.

1 Introduction

Mathematical education is often associated with increasing skills that are considered useful in modern life [4]. But unfortunately, mathematical education ofen times focuses on teaching small, inner-mathematical abilities and students frequently solve mathematical problems with no regard to aspects of the real world [2]. A common countermeasure is the inclusion of real-world problems into the classroom. This has the additional benefit that the problem itself, as well as its solution are motivating for students. However, meaningful real-world problems also tend to be complex and requires students to spend a lot of time while solving them. To keep the motivation high, pre-structuring and regular, visual feedback are desirable for such a problem-solving process.

In this demo paper, we present EvaWeb, a web application designed to support mathematical modelling in classroom. It enables the simulation of the evacuation of buildings with a grid automaton. This domain was chosen because building evacuations are an established area in which mathematical modelling can be taught (cf. [5]) and where comprehensive technology can be applied (cf. [3]) for automatizing calculations, as well as a visualization of the model, its intermediate states, and results. The goal of EvaWeb is to enable students to work on interesting, real-world problems, while keeping the focus of the learning process on mathematical modeling and mathematical evaluation, rather than dry calculations.

* This research was supported by Deutsche Telekom Stiftung

2 Overview over EvaWeb

EvaWeb is a tool for simulating the evacuation of buildings via a grid automaton.

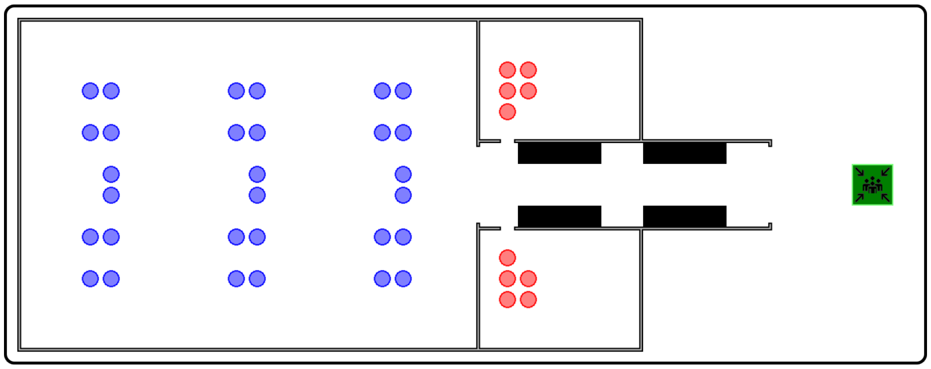


Fig. 1: Example of a scenario depicting a sports hall. The sports hall consists of a large gym area, as well as two dressing rooms. 30 blue and 10 red persons are trying to flee through a hallway, narrowed by four lockers, to the exit.

2.1 Core Functionality

EvaWeb provides the following core functionalities:

1. **Scenario Editor**: Build Scenarios by placing objects in a grid.
2. **Scenario Manager**: Load and store scenarios to a text.
3. **Scenario Player**: Automatically execute scenarios.

In the following section, we will describe each of these functionalities in more details.

Scenario Editor Foremost, the Scenario Editor enables the creation of new scenarios. In this mode, an overview of all available tiles is shown on the right side. These tiles include walls, obstacles, objects, and safe zones.

After selection of a tile on the overview, it is possible to place that tile by clicking on a space on the main map. By default, all tiles in the scenario are empty. Selecting a different tile and placing it at the same position as a prior tile will overwrite this tile.

Additionally, it is possible to place sprites like persons or decorative object (that do not interact with their environment) via the same method. However, sprites will only overwrite other sprite and tiles will only overwrite other tiles. Hence, it is possible that a grid space contains both: a tile, and a sprite.

Using the controls above the tiles and sprites, it is possible to extend or shrink the size of the building.

Scenario Management The Scenario Manager enables loading, storing, and inserting scenarios via two buttons.

When clicking on *Store Scenario to Textbox*, a text representing this scenario is inserted into a textbox in the control area. This text is formatted as `"<spritemap>:<tiles>: <sprites>"`, where `<spritemap>` is the ID of the sprite map used for the tiles and sprites, `<tiles>` is the BASE64-encoded, compressed building plan of all tiles, and `<sprites>` is the BASE64-encoded, compressed information about the sprites in the scenario. The text can then be copied out of the textbox and used as is suitable (e.g., stored to a text file, or transmitted via e-mail). [1]

Afterwards, it can be inserted into the textbox again and loaded using the *Load Scenario from Textbox* button. Additional buttons enable the user to load multiple pre-defined scenarios (including the sports hall depicted in Fig. 1).

Once a scenario String is loaded in the textbox, it is also possible to insert it in the currently loaded scenario. This is done by first hovering over the main map, during which the scenario (after insertion) is pre-shown. With clicking on the Tile, the tiles are inserted. This enables scenario creators to speed up the creation process by re-using certain configuration of tiles (e.g., by inserting multiple rooms with the same layout into a bigger building).

Scenario Player The Scenario Player enables the automatic execution of a scenario. After execution, all persons try to get to one of the safe locations using the algorithm described in Section 2.2. After execution, the results are shown immediately. Most notably, these include:

- **Simulation Steps:** The number of simulation steps until every person arrived at a save location
- **Simulation Micro Steps:** The total number of steps person took in this simulation

As well as the configuration (like algorithms) used to execute the scenario.

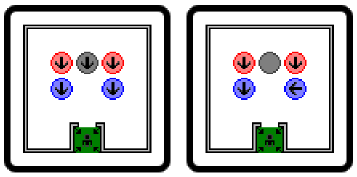
Furthermore, the Simulation Player also enables the visualization of the simulation. Most notably, it is possible to:

- Execute the simulation manually step-by-step or microstep-by-microstep.
- Show the direction each person will be moving to if going to the next step. This direction is shown as an arrow in the circle depicting the person.
- Show the path a person will take to the goal. This path is shown when hovering over a person.
- Jump to the beginning or end of the simulation.

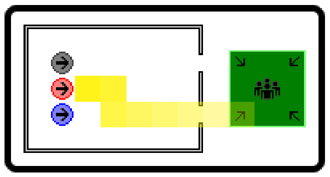
2.2 Evacuation Algorithms

EvaWeb is able to utilize two different algorithms, each of which can be further configured.

[1] The text of the sports hall in Figure 1 consists of 469 characters. This is less than one character for each of the 688 (=13*43) tiles and 40 sprites in the same scenario.



(a) On the left, black moves first and all persons head down. On the right, black is the last one to move. As going left or right would increase the distance to the nearest goal from 3 to 4 (An increase of more than 30%), black chooses not to move at all.



(b) Path highlighting: The arrows highlight the next step a person takes; the yellow overlay highlights the full path the red person takes.

Fig. 2: Snapshots of three simulations using the `default16` spritemap, and the Threshold Accepting strategy ($y = 30\%$, Neumann neighbours) for execution.

Basic Algorithm The basic algorithms for the evacuation is:

```
unmoved_persons = get_all_persons()
do:
    shuffle(unmoved_persons)
    for every person in unmoved_persons:
        desired_choices = person.decide_movement()
        if desired_choices is not empty:
            selected = min(acceptable_option)
            unmoved_persons.remove(person)
while(at least one person moved)
```

Where `min(acceptable_options)` selects one (random) movement option that minimizes the distance to the nearest goal. This, and the shuffling in line two, can lead to randomized results (c.f. Figure 2). The configurable algorithms are derived by further specifying the strategy used in `decide_movement()`.

Strategy 1 (Hillclimbing): In this strategy, each person will only move to a neighbouring tile, if that tile decrease the distance to the nearest safe zone.

Strategy 2 (Threshold Accepting): In this strategy, derived from [1], persons will move to a neighbouring tile, if the minimal distance to any goal after movement is at most $y\%$ greater than the current distance to the nearest goal.

Differences between the strategies Both strategies lead to realistic movements in most scenarios. However, the Threshold Accepting strategy has a "bias for movement" and "incentives" people to switch their goal if the shortest path to the nearest goal is blocked by other people. Hence, it leads to more realistic movements if a scenario makes it likely that there is a jam before the nearest goal. However, using only a single goal, the hillclimbing algorithm leads to more realistic movements as persons are less likely to side-track unnecessarily.

2.3 Configuration

In the Configuration, it is possible to select the algorithm that should be used for execution. The standard algorithm used is the Threshold Accepting algorithm. However, as of right now, it is only possible to select the algorithm itself. It is not (yet) possible to select the parameters used for this algorithm. For example, the Threshold Accepting algorithms is executed with a default value of $y = 30\%$ (that lead to lead to realistic movements for all test scenarios during development).

Additionally, it is possible to define when two grid tiles are neighbours. In the standard configuration, the *Neumann-Neighboruhood* is used: In this neighoburhood, every cell has 4 neighbours (top, right, bottom, left). Alternatively, the *Moore-Neigbhourhood* can be used. In this neighbourhood, every cell has four additional neighours (top-right, bottom-right, bottom-left, top-left).

Lastly, it is also possible to configure the aesthetics and size of the tiles and sprites with different sprite maps.

2.4 Implementation Details

EvaWeb is written in Scala and cross-compiled with ScalaJS to a Javascript file. The repository containing the source code is linked on the homepage. As of the time of submission, the source code consists of around 6500 non-empty lines of code (5700 lines of which are Scala code).

3 Future Work

In this demo paper, we presented EvaWeb, a web application for simulating the evacuation of buildings. As a next step, we want to develop teaching material suitable to teach mathematical modelling for 16-year-olds with EvaWeb. This material will include both the provision of the pre-build buildings, as well as exercise sheets for teachers.

References

1. Dueck, G., Scheuer, T.: Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. Journal of computational physics **90**(1), 161–175 (1990)
2. Greer, B.: Modelling reality in mathematics classrooms: The case of word problems. Learning and instruction **7**(4), 293–307 (1997)
3. Greubel, A., Siller, H.S., Hennecke, M.: Teaching simulation literacy with evacuations. In: European Conference on Technology Enhanced Learning. pp. 200–214. Springer (2020)
4. Jang, H.: Identifying 21st century stem competencies using workplace data. Journal of science education and technology **25**(2), 284–301 (2016)
5. Ruzika, S., Siller, H.S., Bracke, M.: Evakuierungsszenarien in modellierungswochen–ein interessantes und spannendes thema für den mathematikunterricht. In: Neue Materialien für einen realitätsbezogenen Mathematikunterricht 3, pp. 181–190. Springer (2017)