Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Burning Man:
Simulating the behaviour of people in the event of fire

Project Report

Nico Hauser, Andri Horat, Elias Schmid, Jonas Spieler

Zurich
December 2019

Agreement for free-download

We hereby agree to make our source code for this project freely available for download from the web pages of the SOMS chair. Furthermore, we assure that all source code is written by ourselves and is not violating any copyright restrictions.

Nico Hauser

Andri Horat

Elias Schmid

Jonas Spieler

Contents

1 Abstract

This paper describes a model for the behaviour of people in the event of an emergency situation and it's simulation. The model tries to incorporate the general repulsion of people standing too close, the formation of groups in panic situations, the placement of exit signs and many other factors such as different speeds. In order to make observations on the defined model, it is simulated using a physics engine written in javascript that makes it accessible from any place at any time and allows for reproducible and verifiable results. The paper defines a general parameterized model of the environment and the agents which, with the right choice of simulation parameters, may even allow for a safety evaluation of specific settings. Simulations using this model were able to verify results of other papers but also put them into perspective.

2 Introduction and Motivations

This paper was created during the course *Agent-Based Modeling and Social System Simulation* held by Dr. Nino Antulov-Fantulin and Thomas Asikis at ETH Zurich. The goal was to create a project matching the title of the course meaning the modelling of a complex system where humans are the agents. A complex system is in contrast to complicated system not per se hard to implement but rather consists of many simple small parts that on themselves act based on simple rules. The fact that there are a lot of these so-called agents gives rise to behaviours of the whole system that are not always easy to predict or even understand even if the simple rules of a single agent are well-known. As computers keep getting faster and new technology is developed every day, simulations are a good way of understanding these complex systems by playing with the available parameters while looking for emerging patterns and then interpreting the implications for the real world.

3 Description of the Model

For readability this chapter is split as the definition of the environment is to some extent independent of the agent's model.

3.1 Environment model

Before agents can react to a fire an environment has to be created which in our case is a building. The environment model consists of walls, obstacles, doors and signs all of which are objects the agent interacts with. Walls are the most important part as they give the environment a shape and restricts the agent's movement. The same goes for obstacles, the reason why walls and obstacles are separated in our model is that the agent use a raytracing algorithm for finding a path and obstacles such as tables may prevent an agent from moving but do not interfere with their vision. Next are the doors and signs which also have similar roles. As it is the case in real life, safety signs guide an agent to the closest exit where agents are able to escape. Doors have the same effect on agents but represent something different as they're not a physical object but rather represent the agent's memory of where they entered a room. Both signs and doors are directional in our model as both, safety signs and doors, should only be used in one direction when escaping a building.

3.2 Agent model

For the agent model a so-called social force model is used. In short agents behave like particles in newtonian physics. Every agent has a mass, position and velocity and and the intentions and interactions of the agents are represented by forces. The agents are approximated as a circle in the physics engine where each agent has different intrinsic properties that are randomly generated using a normal distribution using a mean and a standard deviation. On one hand the circle radius of the agents is varied representing the different shoulder widths as it was done in [1]. In contrast to this paper, the desired velocity is chosen from a normal distribution as well as this seems more reasonable than all agents having exactly the same desired velocity. In addition we also randomly choose a reaction time that determines how fast an agent does get moving into a given direction.

After the generation of these random values, the agents are a sole product of their environment and their behaviour is based on a set of simple rules.

- Wall repulsion

  To model human behaviour, agents should in general prefer to move away from walls as most humans prefer to have some space around them.

- Agent repulsion

  For the same reason people do not like to stand close to each other, especially not in emergency situations. The closer the distance is, the greater is the applied force on every pair of agents to move away from each other.

- Target attraction

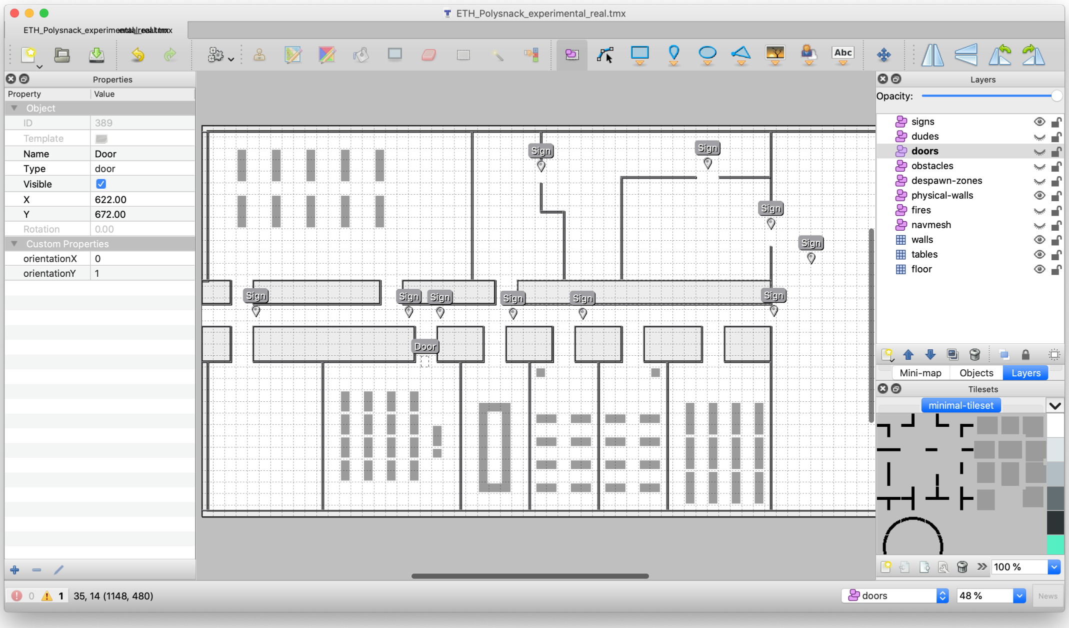
  Last but definitely not least the agents need a task. In the best case, they should try to reach one of the defined escape zones in order to safe themselves. In real life, safety signs guide people to these safe zones so the agents are instructed to always be on the lookout for visible signs guiding them to safety, so we do the same in our model and guide agents to the closest visible safety sign or door. In addition agents will remember what signs and doors they have already passed and won't go into the direction of the same sign or door twice. Should be signs and doors be placed in a strange order, this behaviour ensures that agents won't go into the wrong direction twice. An exception is made to this rule if an agent doesn't have a target anymore. Then they "forget" everything they visited before and start from scratch. This allows for some more complex behaviours such as an agent being pushed back into the room he just was in because of other agents rushing the hallway. If the memory wasn't cleared, the agent now wouldn't leave the room a second time which of course would be a real bad behaviour.

4 Implementation

The implementation of this model is done in javascript using a game library called phaser that already includes a physics engine and has means of drawing the result onto a `HTML` canvas meaning the whole simulation can easily be rendered on a webpage and doesn't require any setup.

4.1 Environment construction

As people are at least to some extent the product of their environment, it is very important that the simulations does not have a fixed environment in which all simulations take place. In fact it is one of the most important parameters when looking at the results, i.e. how many people survived a fire emergency. People in a totally enclosed room have no chance of escaping whereas agents sitting next to an escape zone will have no issues. To make the change of this parameter as accessible as possible, the implementation uses an open source format called `Tiled's map format` (TMX) [2] .



The editing screen of Tiled

Using this editor anyone with basic computer skills is able to adjust the given examples or of course define their own setting. This makes the model much more accessible and allows anyone to make their own tests using this model.

4.2 Agent behaviour

As phaser includes a physics engine we can make use of this by simulating the behaviour of the agents based on forces acting on them where the final movement is the vector addition of all these forces. Phaser has an `update` function that is called every time a new frame is calculated. which is perfectly suited for such a task. The different behavioural rules are implemented as follows

- Wall repulsion

  -

- Agent repulsion

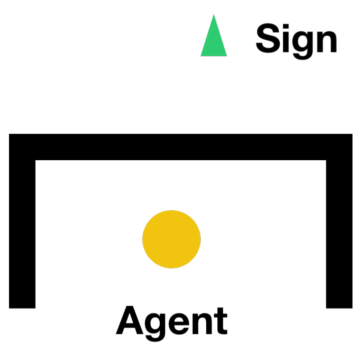
  Agent repulsion can be implemented rather easily: for every pair of agents a force

$$f_r = c_1 \cdot e^{-\frac{d}{c_2}}$$

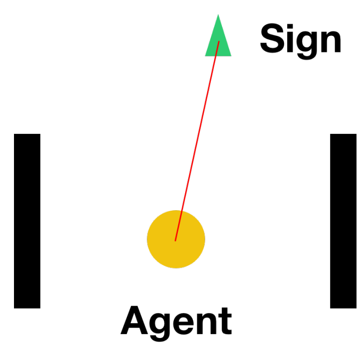
where c_1 and c_2 are adjustable parameters but constant for the two agents and d is the distance between the two agents. As the force is proportional to e^{-d}, the force is exponentially decreasing when linearly increasing the distance, i.e. exponentially increasing when linearly decreasing it. The initial values for c_1 and c_2 were empirically determined and are dependant on phaser's internal units.

- Target attraction

  For checking what signs are visible, we iterate over all of them and use a technique called raytracing. This means we draw a line from the agent's position to the sign's position and check whether this line intersects any wall defined in the environment.



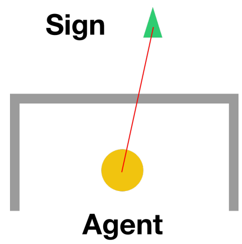
An agent's vision blocked by a wall



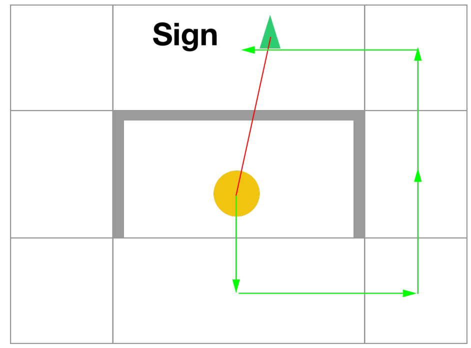
An agent's sight of a sign visualized by a red "ray"

In the first figure above the agent is surrounded by walls meaning a line between the agent and the sign would intersect the front wall and thus in this case the agent wouldn't move towards the sign.

If that's not the case, the sign is indeed visible and we can apply a force into the direction of the sign. This works fine until you introduce obstacles drawn in gray instead of black which stands for a wall. Obstacles collide with the agent as de walls, meaning the agent won't "climb" over it, the difference is that the agent can see "through" or over an obstacle. A good example for obstacles are tables.



Agent stuck within obstacles



Agent boxed in obstacles using pathfinding

Now the issue with directly applying a force to the agent into the direction of the ray becomes obvious, the agent will get stuck as it cannot move through the wall. To circumvent this issue, a so called pathfinding algorithm is used, once the agent sees a sign. Now the agent doesn't just move into the targets direction but rather finds a way to the target and this way is able to walk around obstacles. The pathfinding algorithm works by generating an ordered list of points that the agent can follow to get to the target position.

As soon as we know the target of an agent, we can compute the velocity-correcting force, which tries to accelerate the current physical velocity v_0 of the agent to the desired velocity v. We compute the force by

$$f_t = m \cdot \frac{v_0 - v}{\tau}$$

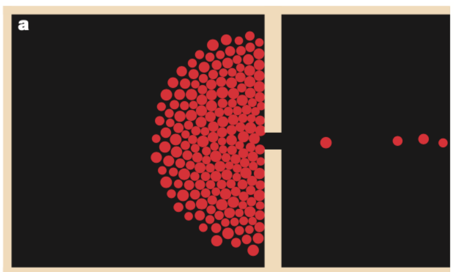
where m and τ are constants, that can be interpreted as the mass and the reaction time of an agent. In our simulation the m is chooses based on the randomly chosen radius and τ is defined per agent and can be used directly.

5 Simulation Results and Discussion

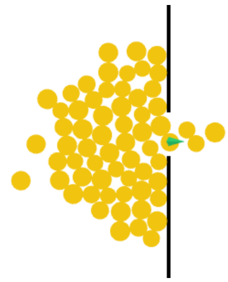
To verify our model we tested our tool against an existing paper. We tried to replicate the experiment of first part of the article *Simulating dynamical features of escape panic* [1]. The second experiment was our main purpose of our model, to simulate the stream of agents located in different rooms of a real building. For this case we took a part of the ETH main building.

5.1 Experiment 1

For the first experiment we replicated the experiments from the paper *Simulating dynamical features of escape panic* from Dirk Helbing, Illes Farkas and Tamas Vicsek in which the agents had to pass a bottleneck in an escape situation. The conclusion was that there is an optimal velocity of about $1.5m/s^-1$ at which all 200 agents left in the least amount of time.



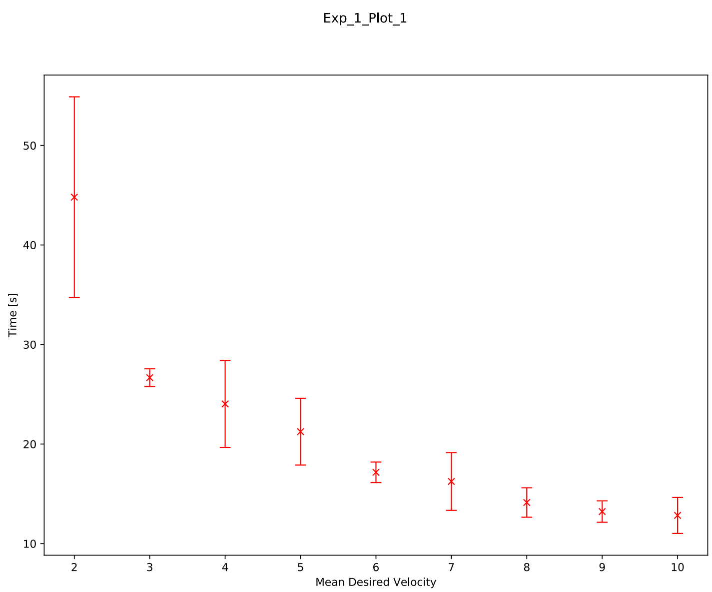
Setup from [1]



This paper's setup

We created a setup similar to that in [1] which is the right image above but in our model this simulation took different turns. When first trying it, some agents started glitching through the wall as the acting forces were extremely high. This is a known issue in physics engines as high forces imply high acceleration and velocity. The path an agent took in a timestep δt can be described as $s_1 = s_0 + v \cdot \delta t$. Since physics engines calculate the new position in discrete time steps without verifying if the path from s_0 to s_1 could have been taken, glitches like the one discribed occur. To circumvent this issue, we on one hand replaced the whole physics engine with a better one that verifies these cases but also reduced the number of agents. This lead to a new issue that suddenly we couldn't replicate the effect from the paper anymore. Higher velocity almost always resulted in better escape times. When adjusting the friction between the agents this behaviour changed again and now most of the time all agents got stuck in a situation similar to the one described in [1] ecept that in our case they were stuck indefinitely. As we cannot create statistics where most of the time everyone gets stuck, it prevented us from having a similar statistic as Helbing, Farkas and Vicsek got. When comparing their results with ours, the differences are obvious. Most likely our model failed to replicate some important behaviour that was considered in [1] but it still seems interesting that the paper describes very exactly all parameters chosen for mass, acceleration time, desired velocity, etc. but only mentions the friction very briefly by stating: "if the friction parameter is large enough"[1]. Depending on how this sentence is interpreted, the results our model produced with a large friction parameter might be more similar than first thought and may also result from a different handling of the friction in the physics engines. For further studies it would be interesting to have a three-dimensional plot with the escape time over the desired velocity and the friction between the agents. Thus the overall question remains, whether a situation visible in the left image above can actually be used in respect to the escape time in real life.

Even tough we could not directly verify the results from [1], the conclusion that higher escape velocities doesn't necessarily result in a shorter escape time could be verified.

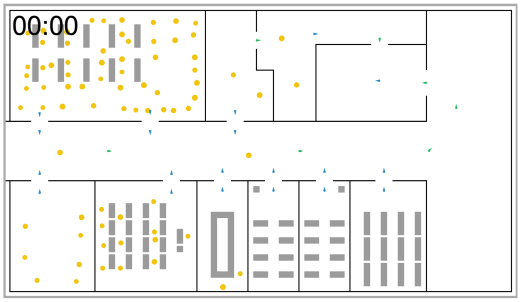


Escape time over mean desired velocity

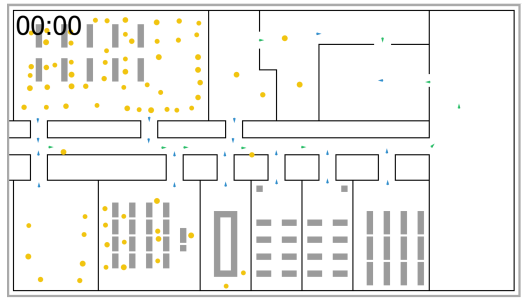
Even tough the escape time continuously shrinks with increasing velocity, the difference keeps getting smaller and hits a minimum. In other words a further increase in the desired velocity won't decrease the escape time as much. This plot is based on five runs per velocity on the described setting.

5.2 Experiment 2

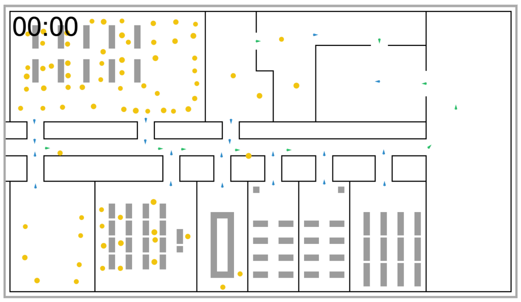
As our model is capable of handling much more complex situations and complex systems are one of the main topics of the course this paper is written in, we thought it might be interesting to test the results of the first experiment in a more complex setting. As the environment we chose a simplified version of the ETH polysnack at noon (1), a modified version with a narrow hallway and a narrow exit (2) and two version in between; one with a narrow hallway and a wide exit (3) and one with a wide hallway and a narrow exit (4).



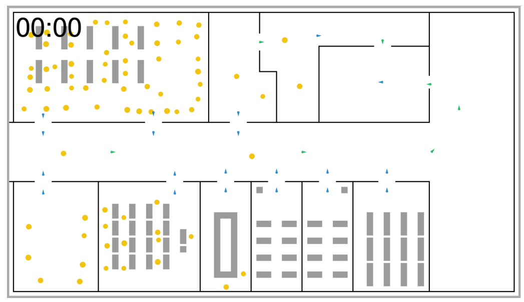
ETH polysnack



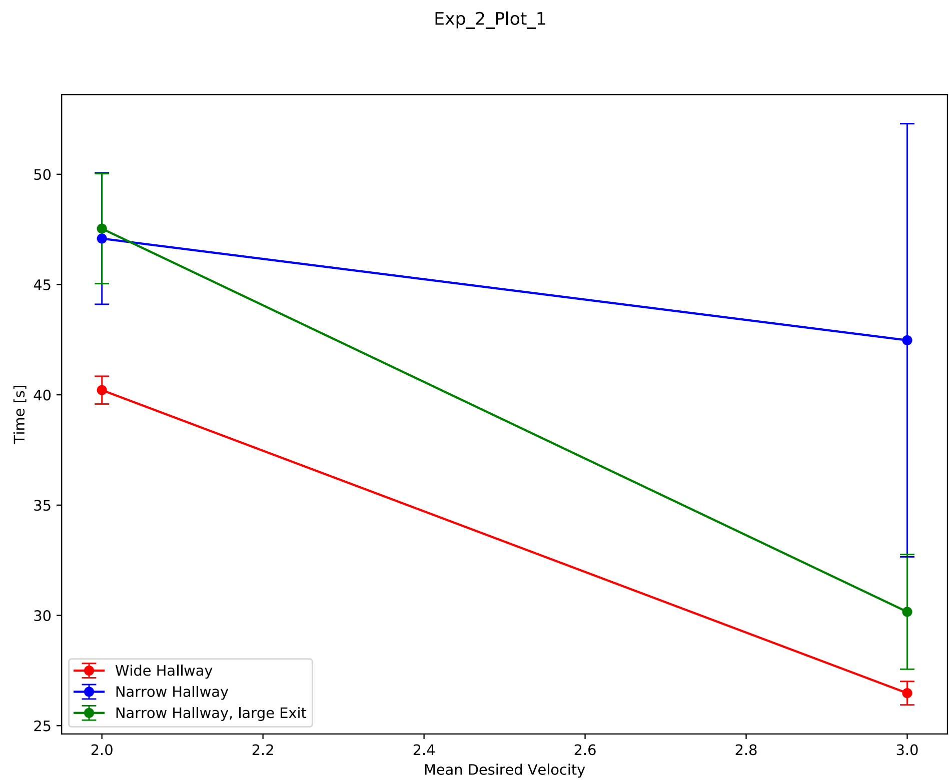
ETH polysnack with a narrow hallway and a narrow exit



ETH polysnack with a narrow hallway and a narrow exit



ETH polysnack with a wide hallway and a narrow exit



Trend of results of different runs in the three settings

As expected the first variation has much lower overall escape time than the second variation dude to the larger exits and wider hallway. What might be interesting is that the third and fourth variations shown in green and x above, are in between the first two. This might be intuitive to some but before running the experiment one could have also guessed that the wide hallway in combination with the narrow exit might lead to some clogging as the throughput suddenly drops. When visually looking at the progress this actually happens but because the hallway is quite long, the time advantage they get from the long hallway with the larger speed more than compensates for the clogging at the exit. When looking at the statistic above, one can observe a large standard deviation for variation (2) which exactly results from this clogging effect that sometimes takes place. As in our model most initial parameters are chosen from a normal distribution, the results vary much more and thus clogging doesn't happen every time. All in all this might not perfectly judge the whole situation correctly as escape time is just one thing but as mention before clogging can result in real bad injuries and is less desirable than being a few seconds faster in a simulation.

5.2.1 Evaluation of the building model

The building model seems pretty solid and wasn't really an issue during our simulations. Also the possibility of having a graphical editor is very handy. Depending on the emergency situation it might be interesting to extend the building model to support fire, smoke or maliciously acting agents such as terrorists. A lot of work was already done in supporting fire, smoke and their impact on sight is accessible when working with the code but it wasn't used in any experiment as the features weren't stable in time.

5.2.2 Evaluation of the agent model

The agent model on the other hand isn't perfect to say the least. It served the simulations pretty well and the graphical results were mostly nice too look at but the social force model has it's limitations. We tried to circumvent some of these limitations with additional raytracing and pathfinding algorithms but the result wasn't perfect. Also there are a lot of undefined parameters in our simulation that weren't fixed to a specific value because of some analysis other than our empirical analysis of what works and what not. The underlying physics engine supports much more that could be made use of but as time was limited it wasn't possible within the scope of this paper. In addition, tracking the forces on the human body and simulating agents falling over would be necessary to model an emergency situation better, especially if clogging takes place. Also the repulsion force between the agents should only apply if the two agents actually see each as otherwise two people standing next to the same wall but in two different rooms will have a pretty strong forces acting on them which of course doesn't represent the reality accurately. For this to be possible, the performance of the raytracing implementation would have to be improved as it would run $O(n^2)$ times every frame (where n is the number of agents). Further improvements would be a force representing the agent's urge to build groups and follow other people. Part of this was already implemented but also wasn't finished in time so it couldn't be used in the final experiments.

6 Summary and Outlook

Even though the experiments didn't yield the results we had hoped for, one can draw interesting conclusion from them and they show where such a social force model reaches its limits. What definitely is different in this project compared to many others is the reproducibility, we really like the idea of having a simulation that can be run right in the web browser and thus the experiments are easily verifiable by others. One mistake we did as a group was to focus too much time on the model itself. As none of us had a lot of experience in building a solid social system simulation, it took us way too long to implement it in a way we were confident with. In the end agents are just are simulated particles in newtonian physics but we wanted to actually build a good model for simulating real people which made us implement everything several times and thus wasting time. Just the map format itself was revised at least ten times! But in conclusion one can say that the modelling and the simulation of this scenario was very fascinating and a lot of fun for everyone involved. Even though many ideas were already mention in the chapter about the model evaluation, we would like to mention that the model could further be improved to support many of the things we mentioned, the resource that was preventing us from doing it was in the end just time. One could start building this simulation more accurately, maybe even in proportions to actual SI units, one could improve the performance of the raytracing algorithm and ensure the agents only act repulsive forces on each other if they actually see each other and one could finish implementing one of the many planned features such as fires, smoke or group formation. But still, we're proud of what we built and hope this project can be used by others and will be further developed.

7 Bibliography

[1] Dirk Helbing, Ills Farkas, and Tams Vicsek. Simulating dynamic features of escape panic. *Nature*, 407:487–490, 09 2000.

[2] Thorbjrn Lindeijer. https://github.com/bjorn/tiled, last visited on 2019-12-01.