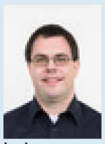
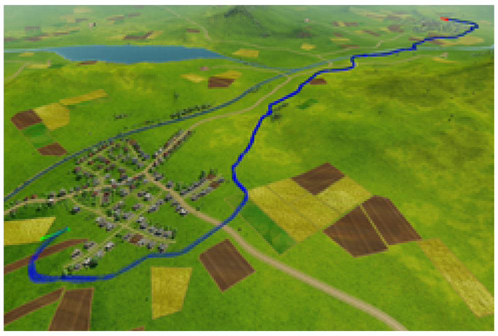
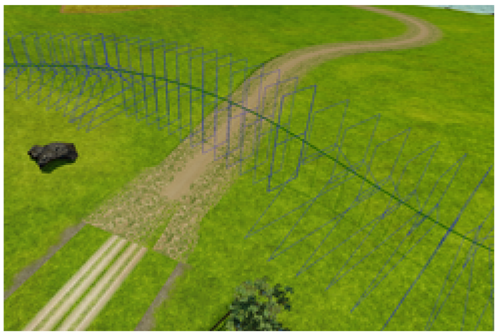

Lukas Kretschmar

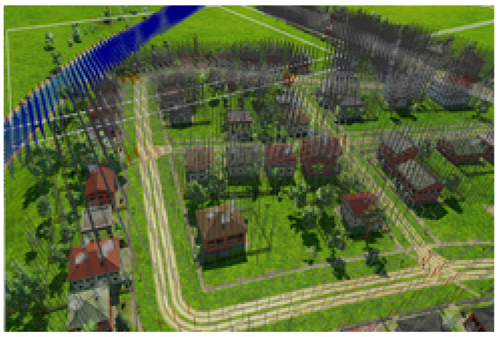
Graduate Candidate	Lukas Kretschmar
Examiner	Prof. Hansjörg Huser
Co-Examiner	Stefan Zettel, Ascentiv AG
Subject Area	Software and Systems
Project Partner	Urban Games GmbH, Schaffhausen, SH

Constrained Pathfinding in a 3D-Environment

Building railroad tracks in Train Fever


A suggested path that connects two cities. The blue shapes indicate the path's bounding box.


The algorithm suggests building a bridge to cross a street rather than using a railroad crossing.


The visualization of the collision detection system. The vertical lines are color-code obstacles. Any obstacle could be dealt with appropriately.

Introduction: As a human player it is very easy to build railroad tracks in a game. We can quickly layout a possible path, see and deal with obstacles and build the track step by step. The goal of this thesis was to develop and test algorithms suggesting reasonable railroad tracks. These tracks were constrained by the game (e.g. curve radius, slope) and, since a map contains obstacles like hills, mountains, rivers, lakes, streets, cities and exposed buildings, the algorithms had to detect collisions and handle them appropriately.

Proceeding: We developed four different implementations of pathfinding algorithms based on the widely known A* algorithm.

- Expanding A*: An extended version of A* based on a dynamically growing graph. The following algorithms are all based on this one.
- Escalating A*: Lowers the resolution until a path is found in a certain amount of time and uses its nodes as waypoints to guide the high resolution search.
- Recursive Escalating A*: Lowers the resolution until a path is found in a certain amount of time, takes the middle and tries to meet there calling itself again.
- TopDown A*: Begins with a low resolution, splits the found path and tries to meet in the middle with a higher resolution until the highest resolution is used.

Additionally, we developed and integrated a collision detection system for the algorithms and defined rules on how to handle obstacles.

Solution: To compare the different approaches, we designed tests exposing the algorithms to different challenges. We also provided a small testing framework to run these tests with multiple configurations to find the best setup and algorithm. At the end, we compared the suggested paths visually and with the measured values (e.g. length, runtime, memory usage, expanded nodes).