# BeamNG.tech Technical Paper

Pascale Maul*, Marc Mueller*, Fabian Enkler*,
Eva Pigova*, Thomas Fischer*, Lefteris Stamatogiannakis*

***Abstract*— Simulation platforms are critical components for the evaluation and development of driver's training applica- tions and intelligent algorithms for advanced driver-assistance systems (ADAS). Driver's training applications are supported by an immersive simulation environment that can easily beadapted by the distributor to match the desired user in- terface and vehicle type. ADAS researchers and developers use BeamNG.tech as a platform for model-in-the-loop testing and development. BeamNG.tech stands out among the several available driving platforms through its custom soft-body physics engine, detailed modeling of the vehicle subsystems, and high degree of adaptability. The soft-body physics approach allows to faithfully simulate the kinematic properties of our vehicle models and leads to a realistic replication of their driving behavior. The modding system allows anyone to customize various components of the simulation, be it the vehicle models, assets, scenarios, or the user interaction. Together with our diverse set of levels, these properties make BeamNG.tech the ideal platform to develop ADAS as well as highly realistic driver's training applications.**

## I. Introduction

Simulations have gained increased interest in the automotive industry and in the training of human drivers. This document illustrates how BeamNG.tech contributes to these two domains to push forward research in the autonomous driving domain and supports driver's training applications.

### A. Simulations in Autonomous Driving

Simulations play a crucial role in the development of autonomous driving systems since they have proven to be a cost- and time-efficient tool. Moreover, ISO/PAS 21448 [1] explicitly proposes a strategy to adopt the use of simulations to minimize residual risk due to insufficient situational awareness of autonomous driving applications. Continuous verification & validation is a key paradigm to acquire sufficient functional safety. An iterative cycle is the foundation of the software development process proposed in ISO/PAS 21448, where the use cases of the respective autonomous driving system (ADS) is repeatedly reviewed to obtain a comprehensive specification of its operative design domain. In this framework, model-in-the-loop (MIL) simulations are a cost- and time-effective tool to implement verification & validation procedures and help develop specifications that lead to safe ADS deployment. But not only ADS development relies on simulations: the performance of any learning-based applications directly depends on the quantity and quality of available data for training and evaluation. Thus, access to large amounts of data is crucial for the successful development of autonomous driving applications, especially if they rely on deep learning.

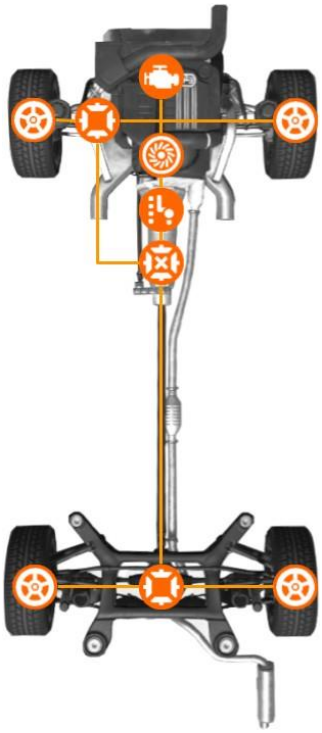

Fig. 1. Schematic representation of the powertrain simulation of the ETK800 854t in BeamNG. The mathematical model of individual compo- nents can be further subdivided but have been omitted in favor of simplicity. The individual components of the powertrain are, from top to bottom andas they are connected: engine, torque converter, gearbox, transfer case thatis connected to the front and rear differential.

### B. Simulations for Driver's Training Applications

The use of simulations in driver's training is a powerful tool to extend classical driving lessons to improve novice drivers' risk awareness. With only a basic gaming PC asequipment, students can safely explore driving situations thatare hard to replicate in the real world or situations that may not necessarily appear during driving lessons. Scenario training allows the students to proceduralise critical skills andto better prepare for driving and traffic challenges. Moreover, this training method allows for a self-guided learning process where students have the opportunity to gain confidence in their driving skills.

* Pascale Maul, Marc Mueller, Fabian Enkler, Eva Pigova, Thomas Fischer, Lefteris Stamatogiannakis BeamNG GmbH, Bremen, Germany {pmaul, mmueller, fenkler, epigova, tfischer, estama}@beamng.gmbh

### C. BeamNG.tech

We present BeamNG.tech as a powerful and versatile simulation platform for the development of autonomous driving systems and driver's training applications. Originating in the entertainment industry, the founders had the vision to develop an authentic and versatile driving simulation. Creating a custom physics engine that is based on soft-body physics principles, and consequently building an authentic vehicle simulation with a faithful replication of true-to- life driving experiences. Today, BeamNG.drive is a popular simulator for realistic driving in the gaming sector where the finely tuned physics model and highly developed graphics create an immersive driving experience in a sandboxed environment that proves to be highly beneficial for driver's training applications. BeamNG.tech expands the capabilities of BeamNG.drive by supporting automated data genera- tion, providing various sensor models that are commonly used in the autonomous driving sector and allowing more parametrization of the overall software. Together with our support for industrial and academic projects, BeamNG.tech proves to be a versatile tool for ADAS development, data generation and a strong foundation for any driver's training application.

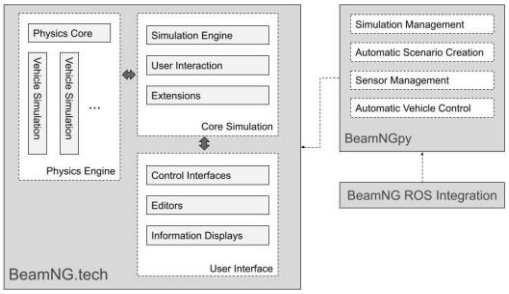


Fig. 2. Architecture of BeamNG.tech and dependencies of provided open-source projects. BeamNG provides three software packages. With BeamNG.tech providing the simulation, BeamNGpy is responsible for sensor management, automated scenario creation and vehicle control, and allows the user to manage the simulation state. BeamNGpy can be usedto integrate BeamNG.tech as a subsystem for X-in-the-loop developmentor other applications. Furthermore, the BeamNG ROS Integration package is available. It depends on the BeamNGpy library and is used to integrate BeamNG.tech into the ROS environment.

## II. Usage

This section describes the individual software components and features of BeamNG.tech that support industrial and academic clients in the development of driver's training and ADAS applications and details the features supporting the respective areas of applications.

### A. Components/Infrastructure

BeamNG GmbH distributes two software packages for academic and industrial use. One is BeamNG.tech, the standalone feature-rich simulation environment. The other one is BeamNGpy – a Python interface targeting ADAS development. While BeamNG.tech targets driver's training applications and provides support for custom content generation, it also provides support for automated data collection that is facilitated by BeamNGpy[1]. Besides its assets, BeamNG.tech consists mainly of C++ and Lua code to manage the simulation environment, while the user interface relies on the Chrome Embedded Framework. Except for the C++ part, the code base is freely accessible to the user and can be adjusted to match the respective product requirements.

### B. BeamNG.tech for driver's training

*1) Vehicle Models:* The vehicle models of BeamNG.tech are composed of three interacting parts that define appearance, kinematic properties, and the propulsion mechanism. BeamNG.tech provides 28 vehicles and 7 trailer models from 13 different brands. While the appearance is defined through a typical 3D model, it also has a structural counterpart defining the physical properties of the vehicle. This physics skeleton predominantly determines the vehicle's interaction

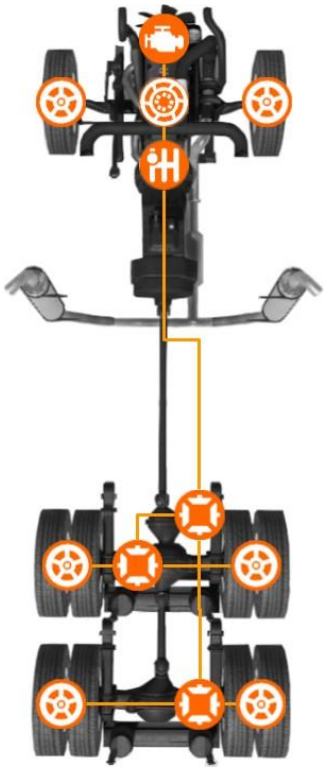

Fig. 3. Schematic representation of the powertrain simulation of the Gravil T65 in BeamNG. The mathematical model of individual components can be further subdivided but have been omitted in favor of simplicity. The individual components of the powertrain are, from top to bottom and as they are connected: engine, clutch, gearbox, first differential that distributes the power between the two-wheel pairs and the consecutive differential that distributes the power within the individual wheel pairs.

[1]https://github.com/BeamNG/BeamNGpy

Fig. 4. Schematic representation of the Tograc qE's powertrain simulation in BeamNG. The mathematical model of individual components can be further subdivided but have been omitted in favor of simplicity. This diagram shows that it's also possible to include two different power sources into the powertrain simulation. The two powertrain trees each consist of an electric motor that is connected to a differential, which is connected to the front/back wheels.

with its environment. It is accompanied by a powertrain simulation that converts steering commands into torque that is applied to the physics skeleton. The individual components of the powertrain do not have any physical properties, they are part of a separate and very detailed simulation that keeps track of its mechanical and electrical components state. BeamNG.tech's powertrain system can be assembled in a plug and play manner and can thus be adapted to fit any configuration from a combustion engine powered vehicle to electric cars, to trucks with four-wheel drive. To demonstrate the flexibility of this system we have included three example powertrain configurations. See Figures 1, 3, and 4.

*2) Scenarios:* Scenarios are used to specify location, environmental conditions, traffic participants and setting up constraints to be fulfilled by the player. They can be used to monitor driving behavior (e.g. by marking a scenario as failed if the player does not respect the speed limit) or to train specific skills (e.g. parking backwards). BeamNG.tech contains 19 levels of whom three levels provide urban environments. The Flowgraph editor is a visual programming tool that supports scenario creation without the need to code. Here a scenario is created by defining a visual graph whose nodes define a certain behavior and whose links connect the different nodes' in and output. With its help, an initial setup can easily be defined as well as different constraints and events to guide the player's behavior. For a Flowgraph example the reader is referred to Figure 5.

### C. BeamNG.tech for ADAS development

*1) Sensor Models:* These commonly used sensors in autonomous driving are available:

- Camera
- Lidar
- IMU
- Ultrasonic

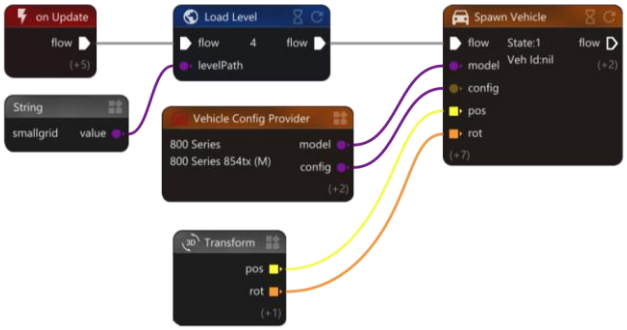


Fig. 5. This figure shows a Flowgraph example for creating a simple scenario, where a vehicle is spawned on the Smallgrid level. There exist different types of nodes, that can be categorized into those that perform a certain task and those that define values. While the main purpose of the latterones is to provide inputs for other nodes, nodes that perform tasks execute commands and rely on the concept of flow to do so. With the help of Flow the order of execution in the network is defined. It is produced exclusively by 'on Update' nodes and is propagated by task performing nodes. In this example the 'String' node is used to define the level, the 'Vehicle Config Provider' selects BeamNG's ETK800 854tx (M), and the 'Transform' node specifies the vehicle's position and orientation. These values are used bythe 'Load Level' and 'Spawn Vehicle' nodes as inputs to perform their respective tasks. The 'on Update' node forwards its flow through the white colored link to the 'Load Level' node which in turn propagates it to the 'Spawn Vehicle' node, thus defining the order of execution.

Fig. 6. Scene from BeamNG.tech's West Coast USA level, automatically generated with the help of BeamNGpy.

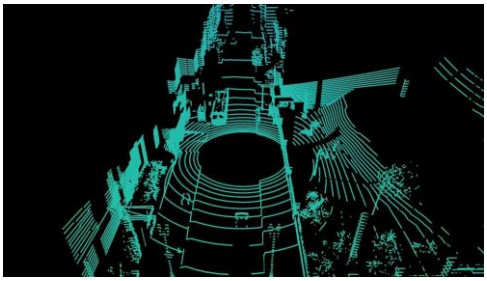

Fig. 7. Visualization of the corresponding Lidar data, captured simultaneously with the image data shown in Figure 5.

- Electrics
- State

The camera sensor provides RGB and depth images, as well as ground truths in form of pixel-wise class annotations, instance annotations, and bounding boxes. These sensors can be customized to fit the individual needs of the users. The electrics sensor represents an easy way to access the vehicle systems internal state. Many values are available, among others: information about the clutch state, engine state, or just whether the turn signal is active. In contrast the state sensor acts similar to a GPS sensor: it contains information about the vehicle's position and orientation in the current map. Our simulation furthermore allows to record information that is otherwise difficult to acquire and quantify in the real world:

- g-force and
- damage data.

Vehicles in BeamNG.tech are represented through a structural graph where nodes have a mass and interact with each other through the edges (beams) that act like springs with varying degrees of stiffness and strength. This approach allows us to measure the stress on individual beams, whether they are broken or not, to measure forces at differentpoints in the vehicle, physical stress, and damage.

*2) Data Acquisition:* Data acquisition plays a central role in the training of learning-based algorithms, especiallyin the one of deep learning systems. Our scenario-basedapproach allows to systematically collect data in each operational design domain. Scenarios help defining a set of conditions, such as traffic participants, the environment, and a range of environmental conditions. All vehicles can be controlled directly through the BeamNGpy library, or the simulation's AI. The library furthermore allows to define a set of custom sensors for each vehicle, whose data can be retrieved on demand. While BeamNGpy plays a substantial role in the automation of BeamNG.tech, it also helps integrat-ing the application into X-in-the-loop settings. Furthermore BeamNG provides the BeamNG ROS Integration, an open-source package that allows to integrate BeamNG.tech into the ROS platform.

*3) Content Creation:* BeamNG.tech and BeamNGpy offer each tool for content creation. A world editor is part of the BeamNG.tech distribution that allows to manually design new levels. BeamNGpy contains functions that support the automatic creation of roads and placement of pre-defined objects.

## III. Theoretical Background

A vehicle's driving behavior is determined through a range of factors. The most obvious one is the driving input defined by steering, braking, acceleration, and state of the gear system. But environmental factors, such as the road surface, inclination, aerodynamic forces play an equally crucial role. Since the choice of the kinematic model is the backbone of BeamNG.tech, this section is dedicated to describing the soft-body physics approach in contrast to the rigid-body approach.

### A. Rigid-Body Physics

This framework is an approach widely adapted in the simulation of cyber physical systems. Rigid-body physics rely on the fundamental assumption that applied forces do not deform objects. One rigid body is characterized through its center of gravity and a finite number of node- points. These object's physical state is described throughtheir spatial rotation and angular velocity. This model canbe easily extended to the popular multi-body approach. Here, individual rigid bodies are assembled into a kinematic chain where individual bodies are connected to each other through mathematical models of joints. The decompositionof one object, in this case vehicle model, into multiple components helps to acquire a more faithful model of the driving dynamics. For example, external force applied to a wheel have a different impact onto the vehicle than the same force applied to the vehicle body. This approach is the fundamental principle that many simulations rely on.

### B. Soft-Body Physics

In contrast, under a soft-body dynamics framework, all individual objects are deformable. BeamNG.tech implements this framework using the spring-mass model. Here, individual objects consist of a 3D graph or physics skeleton. This allows to model the kinematic properties of vehicles in a bottom-up approach, since the overall dynamics of an object are dictated by the state of its individual components. In the terminology used by BeamNG, springs are called *beams* and mass points are called *nodes*. Nodes have a position in space that is updated dependent on the net forces acting on them and influences the state of the beams. BeamNG.tech beams are similar to beams in mechanical engineering, which are structural elements that withstand load primarily by resisting compression or expansion: They have a length that changes depending on the stress forces acting on them and damping forces due to the compressive/expansive rate ofchange. The change in length of a beam will from here onbe called deflection. Deflection can be *elastic*, i.e. the beam will be restored to its original rest length in the absenceof force, or *plastic*, i.e. the deformation will change therest length of the beam, and it can break the beam. Broken beams no longer influence the position of the nodes they are connecting. Beams have deformation and break thresholds that are used to mimic the properties of different materials. The deformation threshold determines how much force can act on the beam before it shows a plastic (i.e. non-reversible)deformation. In turn the break threshold determines howmuch force can act on the beam before it breaks. To mimic the properties of a sponge-like material, one would set alow deformation threshold and a high break threshold, sincesponges are easy to deform but difficult to break. To modela brittle material like glass one would choose a deformation threshold that is higher than the break threshold to avoida deformation of the beam before it breaks. BeamNG.tech is capable of running the simulation in real-time and faster. The base real-time simulation frequency of BeamNG physicscore is 2Khz and it is fixed for the entertainment version

BeamNG.drive. In BeamNG.tech, the simulation frequency can be adjusted to be higher or lower, depending on the user's needs.

## IV. Uses in Industry and Academia

This section presents two publications that used BeamNG.tech for their research projects. While one focuses on the evaluation of ADAS applications, the other one explores the possibilities of damage evaluation with BeamNG.tech.

### *A. BeamNG.tech for the Evaluation of Lane-Keeping Systems*

With the increasing demand for ADAS fueled applications, the testing and evaluation of their subsystems is crucial.Thus, in 2019 Gambi et al., devised AsFault, a system for search-based testing and procedural content generation [2]. The goal of this project was to develop a system that evaluates the lane-keeping capabilities of an AI by designing increasingly challenging road networks. AsFault is initializedwith a set of random road networks. Each road in this network is defined through road segments that are combined with each other to form a road network. One challenge in the procedural generation of roads is the definition of *valid roads*. Here, valid roads are only those whose segments do not have a gap between them and do not self-intersect. After successfully generating a road network, it is instantiated as a scenario in the simulation. The fitness of the road network asa test is then assessed by letting the autonomous agent followa predefined path and periodically measuring the distance between the vehicle position and the lane center. Makinguse of the principles of genetic algorithms, the road network is then incrementally mutated to include more and morechallenging cases. The evaluation of AsFault showed that BeamNG.tech can generate effective test suites by exposing safety-critical problems related to the lane-keeping problem.

### *B. BeamNG.tech for Damage Evaluation*

The soft-body physics model of BeamNG.tech allows for a detailed modeling of stress and deformation on vehicle parts. Together with HDI[2], BeamNG collaborated on a pilot project to generate an image data set for damage evaluation. What is otherwise difficult to measure, can be easily computed in a simulation that keeps track of a score to measure the stress of the individual vehicle parts. BeamNG provided the necessary framework to generate that data. It is publicly available on GitHub as the Impact Data Generation[3] project and creates image data by auto-generating five types of crashes: t-bone, rear end, frontal impact, pole crash, and a no crash scenario. HDI in turn proved that this synthetic data can indeed be used to train a deep learning system to predict whether individual vehicle parts are deformed or not [3]. To increase variability of the data set, vehicle configuration (color and asset selection) and simulation settings (weather, time ofthe day) where randomized. For each crash, images from various perspectives were collected with the ground truth images and damage data. The ground truth here contains a semantic segmentation of the vehicle that annotates vehicle parts such as the different bumpers and fenders, the hitch, radiator, hood, roof and individual wheels. The damage data comprises detailed information regarding the beams, among others the maximum deformation observed, the percentage of deformed and percentage of broken beams etc. With this data the deep learning system was trained to classify vehicle parts and to identify the amount of damage they received. The challenges of this project were the difficulty for the CV system to transfer the knowledge acquired to images showing real crashes. More work is needed to close the domain gap between real crash images and the ones generated through BeamNG.tech. Another challenge is to convert the damage data from the simulation into a meaningful quantity thatcan be used in the real world. In summary, this project isa successful and creative approach in automating damage evaluation.

[2] www.hdi.de

## V. Future Directions

In the future our main focus lies in implementing more features to support driving simulators and AD developers. To do so we are also planning to interface with other frameworks to facilitate scenario creation, the import of real-world maps, traffic management and pedestrians. We will also extend our support for co-simulation with other platforms and robotic systems.

## References

[1] I. O. for Standardization. Road vehicles — safety of the intended functionality.

[2] A. Gambi, M. Mueller, and G. Fraser. *Automatically testing self-driving cars with search-based procedural content generation*. Pages: 328.

[3] P. Maul, P. Godejohann, M. Mueller, E. Pigova, L. Stamatogiannakis, and T. Fischer. Applied sim-to-real transfer for damage estimation.