Transforming Terrain Databases into Battlefield Environments Using Compile-Time Dynamics

Remco van der Meer, Ewan Demeur, Ruben Smelik
TNO Defense, Safety & Security
The Hague, The Netherlands
remco.vandermeer@tno.nl, ewan.demeur@tno.nl, ruben.smelik@tno.nl

ABSTRACT

Modern military operations take place in a dynamic battlefield in which the landscape and infrastructure can change significantly due to weather effects or tactical destruction. Dynamic changes to the environment can affect the mobility of units, the tactical situation or, depending on their severity, invalidate operational plans altogether. For decades, military simulation systems have featured some form of run-time dynamics, however, even today, the support for this is still limited in scale and scope and implemented in a system-specific manner.

This paper introduces compile-time dynamics, a concept that allows large-scale changes to the environment to be applied as part of the terrain generation process. The main advantage of this approach is that more sophisticated algorithms can be used to generate realistic and vastly altered environments compared to what would be possible at run-time. This allows simulation-based training scenarios to take place in environments that match actual battlefields instead of in static, sterile and pristine terrain databases. Furthermore, by integrating the dynamics in the terrain generation process, correlated terrain models can be exported for different simulation systems without requiring specific modifications for each simulator. The approach leverages procedural generation techniques, thereby avoiding the need for costly and cumbersome manually editing tasks.

We illustrate the potential of our approach with a range of natural effects such as seasonal changes and natural disasters such as flooding, combat engineering, and damage to infrastructure due to munition detonations. We show that these changes are not limited to visuals but also affect line of sight and behavior of Computer Generated Forces (CGF), specifically path planning. Finally, we discuss how the approach could be extended from the training domain to mission preparation, where feedback from reconnaissance units in the field could be looped back in the compile-time dynamics pipeline, resulting in an updated virtual representation of the mission area.

ABOUT THE AUTHORS

Remco van der Meer is a systems engineer at TNO in the Netherlands. Remco has been contributing to various parts of our virtual mission environment development suite, amongst which the techniques addressed in this study. He holds a MSc degree in Applied Mathematics and a BSc degree in Applied Physics from Delft University of Technology.

Ewan Demeur is a scientist innovator at TNO in the Netherlands. He holds a MSc degree in artificial intelligence from Maastricht University. His current focus is on modelling 3D terrains and researching new uses for emerging artificial intelligence technologies in the application of terrain generation.

Ruben Smelik is a senior scientist at TNO in the Netherlands. He holds a MSc degree in computer science from Twente University. He earned a PhD degree from Delft University of Technology based on his thesis on the automatic creation of 3D virtual worlds. His current work focuses on innovations in the field of automated synthetic environment modelling for military simulation applications.

Transforming Terrain Databases into Battlefield Environments Using Compile-Time Dynamics

Remco van der Meer, Ewan Demeur, Ruben Smelik
TNO Defense, Safety & Security
The Hague, The Netherlands
remco.vandermeer@tno.nl, ewan.demeur@tno.nl, ruben.smelik@tno.nl

INTRODUCTION

Military operations occur in environments characterized by constant change. Combat activities and damage from munition detonations continuously affect infrastructure and the built environment, potentially making routes planned on maps impractical and altering the tactical situation, such as cover and line-of-sight. Additionally, weather and seasonal effects like heavy rainfall can modify soil conditions and thereby limit off-road mobility for units. Although historic battlefields featured this dynamic nature as well, modern military operations are increasingly often conducted in complex and densely populated urban areas, amplifying the influence of such changes and making it harder to plan effective operations.

It is important that future military leaders are trained to be aware of and able to handle the dynamic and sometimes chaotic nature of the modern battlefield. As simulation-based training becomes a larger part of education and training curricula, and simulation moves into the fields of mission preparation and operational planning, simulations should reflect the battlefield realities as much as possible (see also Pfeiffer & Tamash, 2014). However, as of today, typical simulation terrain databases are anything but dynamic. Instead, they are to a large extent static and immutable, representing a pristine state of the environment. If dynamic effects are supported at all, these effects are often limited in scope and implemented in an ad-hoc, simulator-specific way, which reduces training value and makes the reuse of scenarios or correlated distributed simulation more complicated.

The concept of a dynamic synthetic environment for military simulation is certainly not a novel idea. The earliest publications on this topic go back to the 1990's, where the concept was named dynamic terrain and the focus was on representing terrain deformations such as craters and tire tracks in the surface geometry. This and subsequent approaches had a clear focus on run-time dynamics, i.e., changing aspects of the terrain database during the execution of a simulation scenario. An example of this is switching a 3D building model to a pre-modelled damage state due to a detonation interaction on the DIS or HLA network. The focus on run-time dynamics is logical, as having the environment respond to events in the simulation scenario is an obvious use case. However, the three major limiting factors of run-time dynamics are:

1. Performance constraints limit the extent and quality of the dynamic effects being applied to the terrain and its features. As a result, effects can often only be applied to a small area, e.g., a single building, or must rely on pre-computed states, which might not reflect the specific conditions (e.g., a single pre-modelled building damage state is used for any type of damage, be it due to small-arms fires, artillery shells, or earth quakes).
2. Correlation in distributed simulations with heterogeneous simulation systems is hard to maintain, since each systems implements the change in its own proprietary manner.
3. Run-time dynamics naturally only focusses on changes occurring during the simulation, and cannot describe the state of the terrain at the start of scenario.

This research focusses on the dynamics of the initial state of the terrain. For this, it introduces the concept of *compile-time dynamics*, a method to perform large scale changes to a simulation terrain database in a correlated way. The name is derived from computer programming, where program code is first compiled into a format that can then be executed. Similarly, we compile the application of a set of substantial dynamic changes to an initial, pristine terrain to obtain a new, unique instance of that terrain before the simulation runs.

In our view, this is a novel take on terrain dynamics and it can complement the existing literature on run-time dynamics (a preliminary version of this research was presented at IT^{2}EC2025 (Demeur, Smelik & Kuijper, 2025)). Our objectives are to provide a method to perform changes on a scale that is currently not feasible with traditional dynamic terrain methods, and in a correlated manner, allowing for distributed simulation with heterogeneous systems starting with a correlated dynamic instance of the terrain. We see two main applications of this research:

1. Training in more varied and realistic environments, supporting a wider range of military scenarios, such as operating in an urban environment that has sustained considerable damage due to combat operation or natural disasters, such as flooding.
2. Mission preparation, where a 3D terrain model used for situational awareness and planning operations is updated based on reports in the field. Such a feedback loop can take many forms. A proof-of-concept implementation of field observations reported using a mobile app automatically being fed back into the compile-time dynamics pipeline is presented in this paper.

The approach leverages procedural generation techniques, a broad family of algorithms designed to amplify a small set of input data into simulation-ready content. Thereby, we avoid the need for costly and cumbersome manual editing tasks.

The paper is structured as follows. We first give a brief overview of the previous research on the topic of dynamic terrain and procedural generation. Next, we describe the concept of compile-time dynamics and discuss how it differs from previous approaches. We have made a technological demonstrator of this approach that includes a variety of natural and man-made dynamic effects, the implementation of which are described. Correlated outputs of the dynamic changes are presented. The paper is concluded with an outlook on future work in this topic.

RELATED WORK

We give a brief overview of the body of research on two topics most related to our work: dynamic terrain and procedural content generation. The interested reader is referred to a number of survey papers that go into more detail on these topics.

Dynamic Terrain

As stated in the Introduction, the topic of dynamic terrain has been a research field in the Modelling & Simulation community since the 1990's. Early research focused on run-time computation algorithms for terrain mesh deformation due to explosions and craters and vehicle movement forming tire tracks (Moshell, Blau, Li & Lisle, 1994). A common practice for dynamic changes in distributed simulation is to promote specific terrain objects, such as a target building compound, to be a simulation entity with a (typically pre-modelled) damage state. Of course, the number of terrain objects that can be managed as active entities is limited due to performance and network traffic constraints and the objects need to be selected beforehand, limiting the flexibility of scenarios.

The introduction of web-services and streaming geo-data based terrain databases such as CDB (Open Geospatial Consortium, 2020) and 3D Tiles (Open Geospatial Consortium, 2023) in theory would allow for correlated real-time changes to the environment. If all simulators are able to stream terrain data from a single source, instead of relying on pre-generated terrain databases in proprietary formats, changes to the environment could be pushed to the simulators as an update, preserving the consistency of the state of the environment across the systems. In (Hebert & Sexton, 2012), the authors describe an approach where the data in a CDB is modified at run-time to create, e.g., a crater in the elevation and imagery layers. The connected simulators can pick up these changes and update their 3D terrain surface geometry.

In practice, the adoption of run-time streaming dynamic terrain is slowed down due to competing OGC standards and differences in implementations, the fact that many simulation systems existing today still require a pre-processed proprietary terrain database on disk and there is no standardized approach and message set to notify simulation systems of run-time changes to the synthetic environment.

The NATO MSG task group MSG-156 proposed a similar M&S as a Service (MSaaS) approach for dynamic synthetic environments (NATO STO TR-MSG-156, 2021), where (dynamic) synthetic environment data was streamed from a shared service using generic OGC standards, such as WFS (Open Geospatial Consortium, 2014). Separate, specialized modification services could change the database, e.g., changing weather data or munition detonations. A demonstration of the proposed architecture was implemented by the task group, and a number of simulation systems were able to connect to dynamic SE service (see Gerretsen, Smelik & Smith, 2021, and the accompanying movie clip, NATO STO, 2021).

For a more in-depth analysis of current research on dynamic synthetic environments, showcasing individual algorithms and available tools, we refer to a recent survey (Smelik, van Wermeskerken, Krijnen & Kuijper, 2019).

Procedural Content Generation

Procedural content generation methods and tools form the basis of the implementation of compile-time dynamics. Procedural content generation methods describe a broad range of algorithms that generate some form of content (3D geometry, textures, sound or music clips, …) on the basis of a small set of input parameters. As such, they are also known as data amplification algorithms. They typically combine the input parameters with some rulesets, mathematical tricks or patterns inferred from example data, and together with random number generators, can produce vast amounts of varying content. There are a number of literature surveys available (Smelik, Tutenel, Bidarra & Benes, 2014, and Galin, Guérin, Peytavie, Cordonnier, Cani, Benes, & Gain, 2019).

Procedural generation has been an active research field since the 1980s, stemming from the Computer Graphics research field. Famous examples include Perlin Noise, a versatile noise algorithm used for, e.g., terrain surface generation and Computer Generated Architecture (CGA), which formed the basis for the commercial tool CityEngine (ESRI, 2025). Nowadays, procedural generation techniques have found their way in many entertainment games, movies and tools and middleware. An example of a general-purpose procedural modelling tool is Houdini (Side Effects Software, 2025), which uses hierarchies of nodes and a specialized scripting language to produce 3D models and raytraced renderings, with support for fluid and physics simulations.

AI-based model generation is an upcoming and rapidly developing research field that shows promising initial results. At the time of writing, the quality of the output models is still too low and limited for practical application to terrain database generation. In the near future, however, we expect such AI techniques to provide an avenue complementary to procedural methods for generating 3D models in different dynamic states.

COMPILE-TIME DYNAMICS WORKFLOW

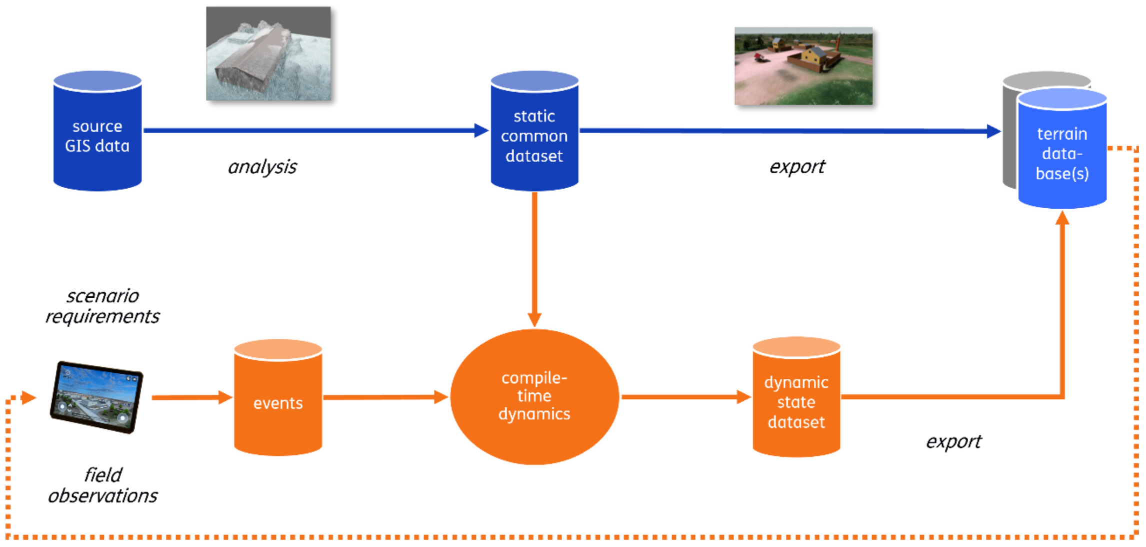


Figure 1. Diagram of the compile-time dynamics workflow

As described in the introduction, this research introduces the concept of *compile-time dynamics*, with the following capabilities:

- Perform large-scale dynamic changes to the synthetic environment on the basis of a state description;
- Support a reach back loop, where observations made in the field trigger the compilation of a new state of the synthetic environment;
- Provide a wide range of heterogenous simulation systems with correlated databases.

The compile-time dynamics workflow is an extension of the traditional workflow for modelling geo-specific, correlated terrain databases from source GIS data. As follows from Figure 1, the typical workflow employed in, e.g., commercial Database Generation Systems (DBGS) is to import all available GIS, analyze, correct and enhance the data to produce a common dataset, and then specialize the common dataset for specific output targets, i.e., terrain databases that adhere to the requirements of specific simulators. For additional information, the typical terrain database generation workflow is described in more detail in the RIEDP SISO Guide (RIEDP Product Development Group, 2018).

The traditional workflow results in a common dataset that is the static representation of the environment. This forms the starting point for generating new dynamic state datasets, resulting in new terrain databases for the export targets. By applying the dynamic changes to the common dataset, we can ensure that the different outputs remain correlated (to the extent possible given simulator-specific constraints). As shown in the figure, compile-time dynamics generates dynamic variants of the common dataset in a non-destructive manner. In fact, many dynamic variants can be generated in parallel given different inputs. The method is fully compatible with simulator-specific run-time dynamics because the generated terrain databases merely contain different data and are functionally identical to the unmodified versions.

Matching our two main application domains, the input can be a set of specifications coming from the requirements of a training scenario (e.g., a geo-specific city under winter conditions and after weeks of intensive combat operations) or actual observations from the mission area. For the latter, an iterative reach back loop is foreseen, where the synthetic environment is updated regularly based on new field reports.

IMPLEMENTATION OF A DEMONSTRATOR

First, we give an overview of the complete workflow, starting from raw source data to exported terrain databases. In our implementation, we built on top of our existing terrain generation architecture called ARA, or Automatic Rapid Analysis, created to efficiently and effectively analyze terrain data and generate correlated terrain databases for several different simulators in use by the NL Defense forces, such as VBS4 and SteelBeastsPro. Using ARA, we generate a baseline dataset, on which we apply our compile-time dynamics method to automatically generate a number of dynamic variants for different simulators. All algorithms described in this section are fully standalone and reusable.

Terrain Dynamics

Typical terrain databases contain several different types of data: raster data describes the terrain elevation, color and surface properties, while vector data describes smaller features that are present in the environment, such as buildings, roads, and vegetation. Since the compile-time dynamics pipeline acts on these terrain databases, algorithms that simulate dynamic effects need to modify one or more of the listed database layers. This section briefly discusses several of these algorithms.

Destruction of the Natural Environment

The natural environment has several elements that might be affected during the course of battle. Most notably, the terrain surface itself can be damaged by events such as munition detonations or the construction of trenches. These affect both the elevation profile as well as the terrain's surface type properties. We implemented a tool that applies these effects using specified or procedurally generated crater and trench locations and their radius. In addition to the terrain itself, vegetation is also affected by these events. Vegetation within the craters or trenches is removed, and vegetation within a certain proximity to the craters are marked as 'damaged' or 'scorched'. This status information is used at database export time to perform model substitution to a pre-modelled damaged or scorched tree, however, it could also be input for a procedural tree model generation algorithm.

Other significant events that may impact vegetation include deforestation and wildfires, both of which were implemented in a similar fashion.

Infrastructure and Building Destruction

Damage to buildings and urban infrastructure such as bridges is a common consequence of military operations, and accurately assessing and modelling its impact is crucial for effective planning and execution. Utilizing commercially available procedural modeling software, a tool has been developed to automatically apply damage to large urban areas. This tool leverages input data from observed munition detonations to procedurally generate the internal structure and debris of both buildings and bridges. By providing a realistic and detailed representation of the damage, compared to the traditional approach of using pre-modelled damage states, this tool aids military planners in assessing potential impacts and making informed decisions.

To introduce dynamic building damage, the common approach to generate geo-specific building models using procedural extrusion from vector footprints was extended. However, the approach could also be applied to, e.g., hand-modelled building models. All these building models needed to be procedurally updated to reflect the dynamic changes, namely munition damage. For this, Houdini, a node-based procedural modeling tool, was used (Side Effects Software, 2025). Houdini uses node graphs that allow for models to be created and updated using a generic blueprint that can apply to models of different shapes and sizes. Using Houdini, building models were converted into damaged models by leveraging known building geometry, such as building geometry reconstructed from LIDAR source data or extruded building footprints, along with an elevation raster. In the Houdini graph, the building geometry is first given internal structure, transforming the original set of faces into detailed models with walls and multiple floors based on the building's height. 3D impact vector points with a radius are then applied to simulate blasts on the building geometry. The resulting chunks of the building removed by the blast are split and scattered across the terrain using a simple physics simulation. The elevation raster ensures that the debris falls correctly on the terrain, providing a realistic representation of the damage.

The same approach is applied to bridge models, with some modifications. Instead of using existing geometry, bridge models are created in Houdini using a basic blueprint and a linear vector. This allows for the procedural generation of bridge structures, which can then be dynamically updated to reflect damage from munition impacts. The process involves applying 3D impact vector points to simulate blasts, and the resulting debris is scattered across the terrain using a physics simulation, ensuring a realistic representation of the damage. Additionally, the road network is updated if the bridge is heavily damaged, ensuring that any navigation or route planning tools do not use a damaged or destroyed bridge.

Precipitation

Rainfall and snow can have an enormous impact on the tactical situation, as floods or droughts greatly affect the trafficability of terrain and the navigability of rivers and bodies of water. Snow can present a significant risk in the form of avalanches when it accumulates on adequately steep slopes.

We developed two empirical algorithms to simulate the effects of precipitation. One algorithm simulates rain over relatively short timespans of up to several days for a given weather profile. This algorithm takes water runoff (Št'ava, Beneš, Brisbin & Křivánek, 2008), infiltration and evaporation into account and outputs puddles, streams and a water saturation level of the terrain surface. This way, floodings can be simulated.

We created another algorithm to process snowfall and melting over longer periods of time, allowing the simulation of the accumulation and persistence of snow on more mountainous terrains. This algorithm takes the average weather and the solar position into account, and is able to feed the melting of snow into the rain simulator to generate believable streams that vary significantly depending on the date and time of day.

Seasons

Seasonal changes can significantly impact the realism of the terrain being used. A terrain built using data and models taken in summer does not correlate with what we would see in the winter. While there are many changes that can be made to the natural environment as a result of changing seasons, in our current implementation, we switch to season-specific tree and low vegetation models to represent the dynamic season state in the resulting database.

Feedback Loop

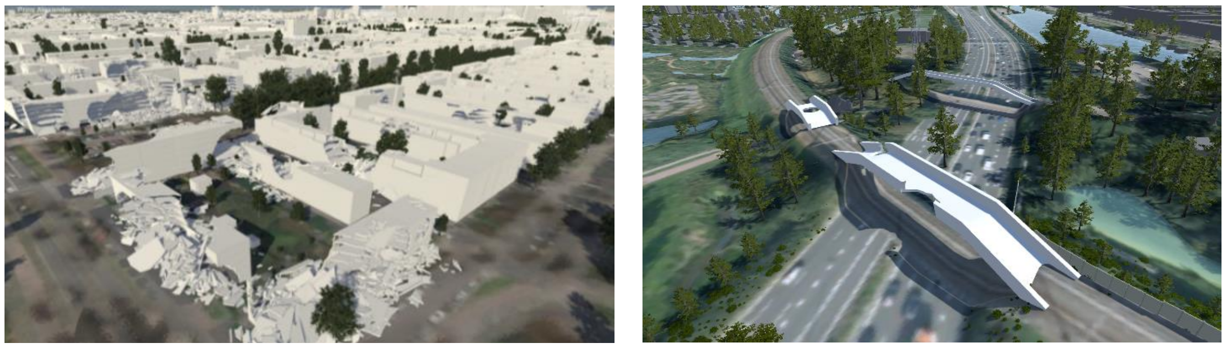
The terrain generation pipeline begins with existing sensor data, which is analyzed and converted into a dataset that is then exported into simulator-specific terrain databases for all relevant simulators. In this research, the pipeline was extended into a loop, where new observations trigger the pipeline to produce an updated version of the terrain model using compile-time dynamics. To gather new observations of the terrain, a prototype was developed to allow users to provide feedback on changes to terrain or features they observe during operations. The prototype feedback functionality was created in an Android-based mobile application designed for mission planning and execution. This allows users to note observations, providing the information needed for the pipeline to compile dynamic changes in the next iteration of the terrain model. The feedback functionality enables users to introduce observational data to the pipeline quickly and efficiently, with enough detail to effectively visualize the changes, ensuring that the next iteration of the terrain model accurately reflects real-world observations.

RESULTS

Dynamic Instances

This section showcases several terrain databases and their variants with compile-time dynamics applied to them. The databases are exported to three simulators: VBS4, SteelBeastsPro4, and the Unity game engine. These databases cover two different regions: the area of surrounding Rotterdam in the Netherlands, which is used to showcase the effects of damage to infrastructure and buildings and rain, and a more mountainous region in Norway, which we use to showcase snowfall, water runoff and seasonal changes.

The environment model of Rotterdam was modified in several different ways to simulate a battlefield scenario. This scenario included the construction of trenches to defend the airport, bombardments at the airport and the infrastructure and buildings close by, and was followed by heavy rainfall.



(a) **(b)**

Figure 2. Damaged buildings (a) and bridges (b) in a terrain model of Rotterdam, the Netherlands

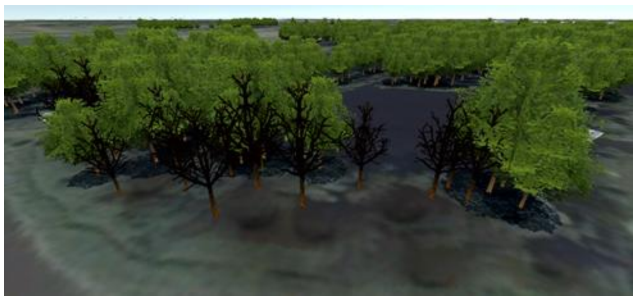


Figure 3. Scorched trees close to craters

The first part of the scenario describes the battle in dry conditions. The scenario explicitly defined the locations of trenches and munition detonations. Figure 2 depicts the damage that was done to buildings (a) and bridges (b) and the debris that was scattered in the area. Figure 3 shows the effects of munition detonations on vegetation; trees at the impact locations were completely removed, and trees in proximity were scorched.

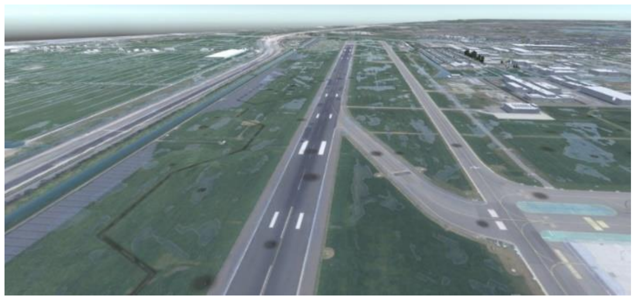


Figure 4. Craters, trenches, and minor flooding near Rotterdam airport

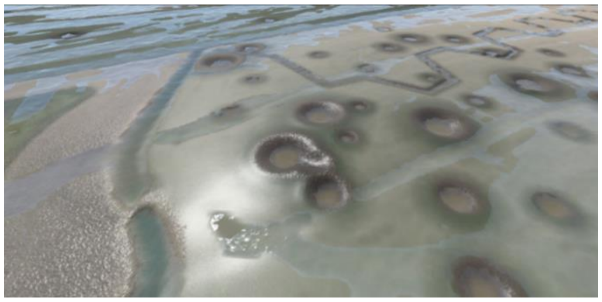


Figure 5. Flooded craters and trenches due to heavy rain

In the second part of the scenario, significant rainfall was simulated over several days. The only input for this part of the scenario, in addition to the events that occurred previously, was a rainy weather profile. Figure 4 shows the construction of trenches near the runway of the airport, impact craters and the formation of puddles due to the heavy rain. Figure 5 showcases that generated craters and trenches can get filled with puddles by the rain simulator.

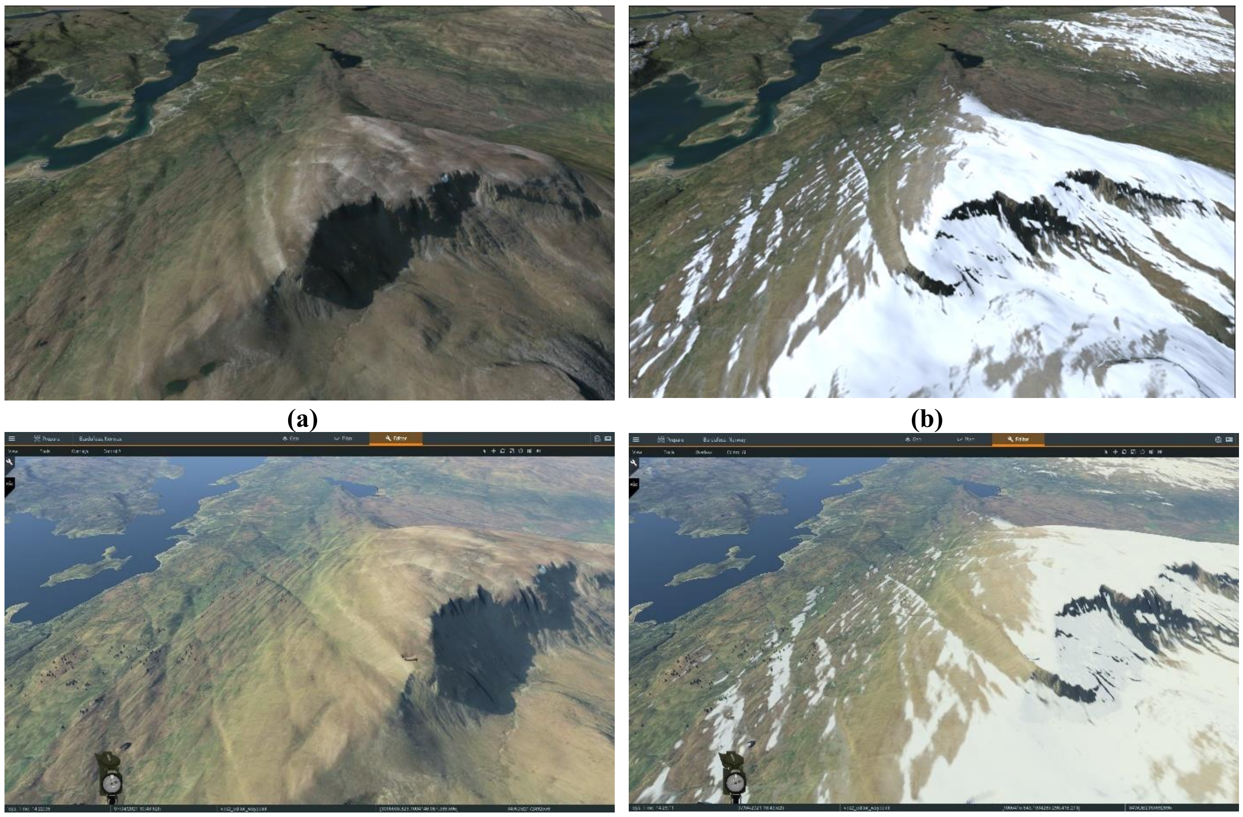

(a) (b) (c) (d)

Figure 6. Norwegian mountains in summer and autumn. Figures a and b show the database in Unity. Figures c and d show the same database in VBS4.

A terrain database in Norway is used to show seasonal changes including precipitation (snowfall, rain), snow persistence and melting and vegetation changes. Dynamic instances are generated at different times of year with specified weather profiles, resulting in different amounts of snow cover, and different vegetation density. Figure 6 shows correlated terrain databases of these dynamic instances in Unity and VBS4.

Simulating the snow for longer periods of time, in particular until the temperatures start increasing again in spring and summer, enables us to simulate streams and puddles as a result of meltwater runoff.

Feedback loop

The feedback functionality was implemented in the Android-based mission and planning application. The user could use lines, spheres, areals, and points to effectively point to changes in the terrain or damage to features.

Figure 7 shows the result of a linear feedback item describing a newly observed trench in the terrain. This feedback is shown using a partially transparent line on the terrain, and a label that describes the type of dynamic change that has occurred. When a user first draws the line on the terrain a window appears asking for additional information on the observation. In the case of a trench, the estimated depth and width of the trench are asked for. The goal of this functionality is to only ask for the most important information so that the input of new observations is not too cumbersome for the user, yet can still allow for an effective visualization of the changes in the following iteration of the terrain.

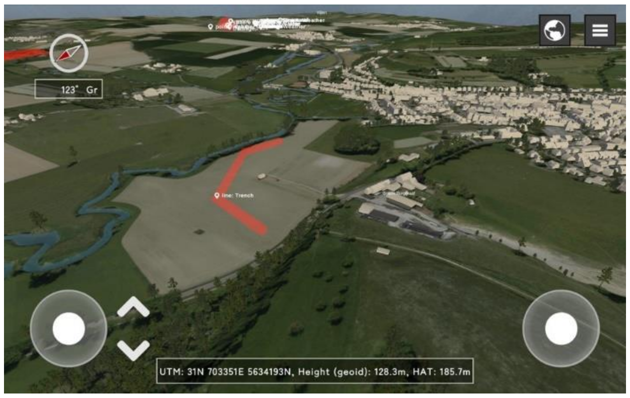

123° Gr
line: Trench
UTM: 31N 703351E 5634193N, Height (geoid): 128.3m, HAT: 185.7m

Figure 7. Feedback functionality, prototyped in a Unity-based mobile app

Performance considerations

Our compile-time dynamics pipeline is able to process complex and large-scale events that significantly alter the environment. However, these algorithms operate on geodata and precede any conversions to simulator-specific data formats. As such, the simulation databases need to be regenerated whenever the scenario describing the dynamic events is modified. The process is therefore much slower than real-time, but can in most cases still run within one to several hours, even for large and high-resolution terrain databases. On the other hand, because the dynamics pipeline runs at compile-time, no expensive processes need to be executed at run-time. For these algorithms, this also enables the use of greater computational resources than what is typically available for most simulation systems, making the method highly scalable.

Where the compile-time dynamics workflow excels, is in the labor time savings it can provide. The procedural nature of these algorithms makes it possible to generate believable scenarios with very little input in a controllable way. That makes this workflow well suited for training purposes, as the scenarios can be easily tailored to fit training objectives. For example, the only inputs required to generate the snowy conditions shown in Figure 6 and the floods shown in Figure 4 and 5, are a timespan to simulate and the weather pattern for that time window.

CONCLUSION

Synthetic environments for military training and mission preparation do not fully capture the dynamic nature of modern battlefields. They tend to provide static and pristine representations of the real world. As such, they are still a long way from being accurate and up to date to the level required for advanced applications such as AI-based Course of Action analysis or digital twinning.

Previous research focused on run-time dynamics, providing algorithms for deforming or destroying individual terrain objects during the execution of a simulation, such as buildings or the terrain surface. This research instead focusses on the dynamics of the initial state of the synthetic environment. For this, it introduces the concept of compile-time dynamics, a method to perform large scale changes to simulation terrain databases in a correlated way.

A prototype of compile-time dynamics was built upon an existing terrain generation software architecture. To showcase the breadth of dynamic changes that can be applied to a synthetic environment, several procedural and simulation-based algorithms were implemented. Examples include soil infiltration and runoff of precipitation and snow accumulation for seasonal changes and flooding caused by excessive rainfall, and destruction of the environment due to munition detonations. The examples highlight the fact that very different dynamic states can be generated from a single geo-specific environment in a non-destructive manner, and the new possibilities for scenario development that the capability of applying large-scale changes to the synthetic environment provides, all of which are currently infeasible to perform at run-time.

We believe that this research is a step towards synthetic environments that better match the reality of modern battlefields. However, there are many more research challenges to overcome to make a true digital twin of tomorrow's battlefields feasible. Avenues of future work include:

- Expanding the set of dynamic effects supported with natural disasters, such as earth quakes, and combat engineering operations, and support procedural destruction of additional feature types, such as power- and communication infrastructure;
- Updating information layers for steering CGF behavior and analysis models to reflect changes in, e.g., navigation, cover, and infrastructure connectivity;
- Implementing and testing the feedback loop in a real-world scenario;
- Matching parametrized procedural destruction models to sensor data, such as point-cloud data captured of partially destroyed buildings or infrastructure.

In conclusion, the introduction of compile-time dynamics for synthetic environments provides a solid base for creating more realistic and adaptable military training scenarios and provides ample opportunities for extension.

REFERENCES

Demeur, E., Smelik, R.M., Kuijper, F. (2025). Bringing Battlefield Reality to the Simulation Scene: Compile-time Terrain Dynamics. Presentation at IT^{2}EC 2025 (Oslo, Norway).

ESRI (2025). CityEngine. Available from https://www.esri.com/en-us/arcgis/products/arcgis-cityengine

Gerretsen, A., Smelik, R.M., Smith, N. (2021). MSaaS based architecture for Dynamic Synthetic Environments in Distributed Simulations. SISO Simulation Innovation Workshop 2021.

Galin, E., Guérin, E., Peytavie, A., Cordonnier, G., Cani, M., Beneš, B., Gain, J. (2019). A Review of Digital Terrain Modeling. Computer Graphics Forum, 38(2), 553-577. https://doi.org/10.1111/cgf.13657

Hebert, K.J., & Sexton, D. (2012). Dynamic Synthetic Environments Through Run-Time Modification of Source Data. In Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC 2012).

Kuijper, F. (2018). Building Virtual Training Areas, Challenges and Solutions. Presentation at IT^{2}EC 2018, Stuttgart, Germany.

Moshell, J.M., Blau, B., Li, X., Lisle, C.. (1994). Dynamic Terrain. Simulation, 62(1), 29–40.

NATO STO TR-MSG-156 (2021). Correlated Dynamic Synthetic Environments for Distributed Simulation: Final Report of MSG-156.

NATO STO (2021). Dynamic Synthetic Environments for Distributed Simulation (movie). Available from https://www.youtube.com/watch?v=Gv6k7kqI6Es

Open Geospatial Consortium (2020). OGC CDB 1.2. Available from http://www.opengis.net/doc/IS/CDB-core/1.2

Open Geospatial Consortium (2023). OGC 3D Tiles 1.1. Available from http://www.opengis.net/doc/cs/3D-Tiles/1.1

Open Geospatial Consortium (2014). Web Feature Service 2.0.

Pfeiffer, K.D. & Tamash, T. (2014). Measuring the Impact of Natural Environment Representation on Combat Simulation Outcomes. In Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC 2014).

RIEDP Product Development Group (2018). Reuse and Interoperation of Environmental Data and Processes (RIEDP) Data Model Foundations (SISO-GUIDE007-2018 v1.4).

Side Effects Software (2025). Houdini FX. Available from https://www.sidefx.com/products/houdini

Smelik, R.M., Kuijper, F., Nusman, D., Liberg, D. (2017). Virtual Training Areas for Vertically Integrated Simulation Exercises of Land Forces. Presentation at IT^{2}EC 2017, Rotterdam, The Netherlands.

Smelik, R.M., Tutenel, T., Bidarra, R., Beneš, B. (2014). A Survey on Procedural Methods for Virtual Worlds. Computer Graphics Forum, 33(6), 31-50. https://doi.org/10.1111/cgf.12276

Smelik, R.M., van Wermeskerken, F., Krijnen, R., & Kuijper, F. (2019). Dynamic Synthetic Environments: a Survey. The Journal of Defense Modeling and Simulation, 16(3), 255–271. https://doi.org/10.1177/1548512918811954

Šť'ava, O., Beneš, B., Brisbin, M., & Křivánek, J. (2008). Interactive Terrain Modeling using Hydraulic Erosion. In Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '08).