

Research on an Open Source Physical Simulator for the Autonomous Construction Machinery Development

Daisuke Endo¹, Yosuke Matsusaka¹, Genki Yamauchi¹ and Takeshi Hashimoto¹

¹Public Works Research Institute, Japan

endou-d177cl@pwri.go.jp, matsusaka@mid-japan.com, yamauchi-g573bs@pwri.go.jp, t-hashimoto@pwri.go.jp

Abstract –

Facing a labor shortage, the civil construction industry is increasingly focused on automating machinery. The Public Works Research Institute in Japan has introduced 'OPERA,' an open development platform, to encourage stakeholder participation in areas like IT, robotics, and AI. Key to OPERA is its open-source software for model-based autonomous construction. Currently, it offers two physical simulators with distinct physics engines, enabling user-specific applications. This paper covers their technical differences, and current states.

Keywords –

Autonomous Construction; Dynamics Simulator; OPERA;

1 Introduction

To support technical research and development, the Public Works Research Institute is developing OPERA (Open Platform for Earthwork with Robotics and Autonomy), for autonomous construction development [1]. One key component of OPERA is its dynamics simulator, as shown in Figure 1, available on GitHub [2]. This simulator plays a crucial role in the development of automation software in construction machinery by integrating models of both machinery and terrain. It visually demonstrates how terrain reacts to machinery operations and generates numerical data about the conditions of both machinery and terrain. This paper elaborates on the dynamics simulator's technical aspects and discusses the two versions of simulators, each utilizing a different physics engine.

2 Overviews of the Simulator

This chapter introduces the dynamics simulator for research and development of autonomous construction technology provided by OPERA. First, the essential requirements for this simulator are described. In addition, the reasons for selecting the two types of dynamics-engine-based simulator platforms developed to meet these requirements are summarized.

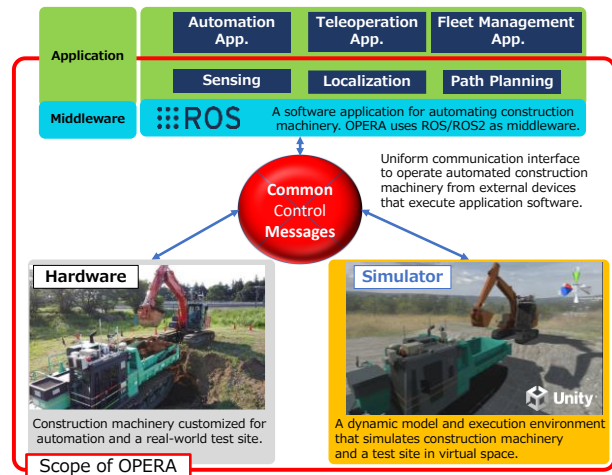


Figure 1. Scope of the open platform OPERA

2.1 Simulator Requirements

The dynamics simulator is designed for developing software that automates construction machinery, aiming to replace real machines and environments with a virtual setup. Its primary objective is to be Open Source Software (OSS) to encourage widespread usage without mandating users to disclose their source code. Additionally, the simulator should be cost-effective and user-friendly[1]. To ensure these, it utilizes existing platforms with robust ecosystems instead of proprietary systems:

The simulator's technical specifications include:

1. Compatibility with ROS [3] and other middleware.
2. 3D rendering to display calculation results.
3. Physical-based behavior simulation of construction machinery.
4. Capability to handle terrain deformation due to interaction with construction machinery.
5. Real-time or faster calculation and rendering speeds.

Concerning these specifications, it's crucial that the simulator is compatible with ROS (Robot Operating System) or ROS2, since these systems are widely utilized in the field of autonomous construction. The

function to interact with other middleware is also important. For terrain interaction, the simulator is required to realistically mimic the way construction machinery alters the ground, as observed in earthwork operations, thereby accurately reflecting the physical transformations.

2.2 Selection for Simulator Platform

As a 3D physical simulator compatible with ROS middleware adopted by OPERA, Gazebo [4] is a widely recognized and freely accessible tool. Given its popularity among ROS users and the established ecosystem it offers, Gazebo is a general choice in this domain.

However, since Gazebo is designed for integration with ROS, it lacks compatibility with other middleware such as OROCOS [5], OpenRTM [6], etc. As previously mentioned, the policy for OPERA's simulator is to make it usable with middleware other than ROS, thus platforms other than Gazebo were also considered.

In recent years, there has been a trend to adopt game engines as platforms for physical simulators for software development to automate cars and robots. Compared to traditional dedicated simulators, they have the following characteristics and advantages, which are the main reasons for their increasing use:

- Availability of abundant 3D rendering functions

and physical engines.

- Well-developed community and support.
- High versatility and extensibility.
- Potential for reduced development costs.

The major game engines adoptable for the simulator platform are Unity [7] and Unreal Engine [8]. Among these two, OPERA adopted Unity, which had a larger developer community and more abundant assets as of March 2021 when the selection was made. This decision was based on the aforementioned policy of making it easy for users to handle the simulator to increase the number of users as much as possible. Table 1 summarizes the results of the study on simulator selection. The criteria for each rating were determined by relative evaluation based on the results of our own preliminary research on each simulator. In conclusion, we decided to prepare two types of simulators with different physics engines, and let users choose which one to use depending on the cost-effectiveness they seek. This paper describes about these two types of simulators in following sections.

3. PhysX Version of the Simulator

This chapter describes the PhysX [11] version of the simulator, focusing on the technical content related to the construction machinery model, the soil ground model, and the interaction between construction machinery and soil ground.

Table 1. Results of platform selection for simulator.

Simulator Platform (Developer)		Gazebo (Open Robotics)		Unity (Unity Technologies)	
Physics Engine (Developer)		Open Dynamics Engine [9] (Russel Smith)	Bullet Physics [10] (Erwin Coumans)	Nvidia PhysX [11] (Nvidia)	AGX Dynamics [12] (Algorix)
①Affinity for middleware	ROS	◎	◎	○	○
	Other	×	×	○	○
②3D rendering availability		○	○	◎	◎
③Motion simulation for machinery		○	○	○	○
④Ground deformation simulation		△~○ (Need to self-make)	△~○ (Need to self-make)	△~○ (Need to self-make)	◎ (Library available)
⑤Real-time Simulation Capability		×	△	○~◎	◎
Cost	Platform	○ : Available for free		○ : Available for free (Depend on Unity's License system.)	
	Physics Engine	○ : Available for free	○ : Available for free	○ : Available for free	× : Paid
Note			Using particle models for sediment representation is difficult for real-time performance.	Implementing soil models and interactions between construction machine and soil requires self-make.	Commercial use cases exist for soil models and their interaction with construction equipment.
Acceptance or Rejection		×	×	△~○	△~○

◎ : Suitable ○ : No Problem
△ : Concerned × : Not Suitable

3.1 Construction Machinery Model

The simulation model of construction machinery was crafted for realistic representation and user customization without infringing manufacturers' intellectual property. The process, shown in Figure 2, included:

1. Capturing the machine's (that is our property) exterior as a 3D point cloud using LiDAR.
2. Disassembling each machine into links to create a 3D mesh model from the point cloud. Joint connections, relative positions, and kinematic parameters were extracted, using both manufacturer catalogs and preliminary surveys.
3. Describing the machine in URDF (Unified Robot Description Format), including weight and other inertia parameters based on measurements, despite some errors.
4. The URDF file was imported into Unity, attaching components for operation as a game object. In the PhysX version, the ArticulationBody component was applied to all links, setting joint types, motion ranges, and dynamics. The tracks were modelled as a six-wheel skid steer utilizing WheelCollider Component, rather than precise tracks.


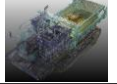



Procedure for Making a Machinery Model	Excavator ZX120 (HITACHI)	Crawler Carrier IC120 (KATO)
1. Obtain 3D point cloud data of PWRI's construction machinery with 3D LiDAR.		
2. Make 3D mesh data based on 3D point cloud data, and then obtain Denavit-Hartenberg(DH) parameters.		
3. Make URDF file, and import the file to Unity.		
4. Add and set configuration of Unity components for each Unity object.		

Figure 2. Making procedure for machinery model.

3.2 Soil Ground Model

In OPERA's simulator, the standard Unity Terrain object is enhanced to simulate terrain deformation due to construction machinery. To model moist soil's viscous behavior efficiently in real-time, a simplified method based on the Discrete Element Method (DEM), inspired by Daniel Holtz et al. [13], is implemented. This model treats soil as particles of radius R , where a cohesion force F acts between particles closer than a distance D . When particles are farther apart, no force acts between them (Figure 3). These parameters can be changed via GUI.

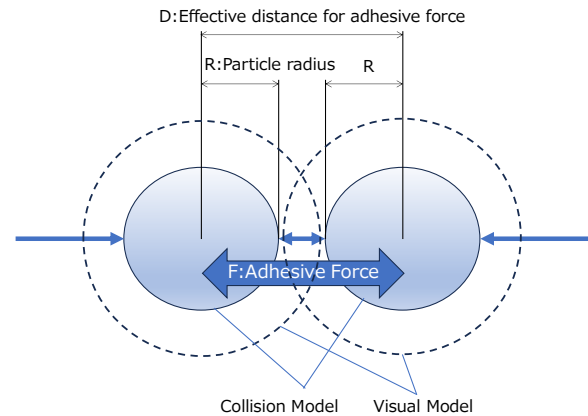


Figure 3. Adhesive force model for soil particles.

3.3 Physical Interaction

At December 2023, the PhysX version doesn't calculate the interaction force between construction machinery and soil ground during collapse or deformation. When parts of the machinery, excluding the hydraulic shovel's bucket, contact the ground, they are treated as rigid bodies in simulation. However, two distinct interference shapes are set for the bucket part:

1. Bucket-Terrain interference detection shape.

This is an interference shape that only detects interference and disables the reaction forces generation with other objects. Particles to represent soil behavior are generated when the bucket contacts the ground.

2. Particle retention interference shape.

This interference shape in the bucket prevents interaction with the ground (Terrain) while effectively managing particles and other objects.

When the bucket overlaps with the Terrain, an equivalent volume is removed, generating particles in the bucket. These interact as rigid bodies and are removed upon contact with the Terrain, increasing its height equivalent to their volume.

This method, a simplified representation of the construction machinery-soil-ground interaction, calculates only the force between particles and the bucket's retention shape. The force at particle generation is impulsive and unstable. Additionally, the feature for simulating terrain deformation through particle creation can be toggled on and off via the GUI.

4. AGX Dynamics Version of the Simulator

This chapter outlines the AGX [12] version of the simulator, highlighting the technical aspects of the construction machinery model, soil ground model, and their interaction. Due to space constraints, it primarily details the differences from PhysX version.

4.1 Construction Machinery Model

In the construction machinery creation process outlined in Section 3.1, the key distinction between the PhysX and AGX versions is in step 4. The AGX version, utilizing the AGXUnity [14] plugin for Unity by AGX Dynamics, attaches various components and sets parameters differently. Unlike the PhysX version, which uses Unity's standard ArticulationBody, the AGX version employs AGXUnity's RigidBody component to define physical properties, joint types, constraints, and dynamics model parameters. The tracks is simulated utilizing AGXUnity's TrackWheel and Track components, with sprockets and idlers modeled by TrackWheel and the track itself by the Track component. The AGX version's track comprises 44 evenly spaced flat plates revolving around the track's outer orbit, though further details are not included here.

4.2 Soil Ground Model

The Unity standard object Terrain has the DeformableTerrain component provided by AGXUnity attached. This synchronizes the shape of AGXTerrain with that of Terrain, and physical property parameters of the soil can be set using this GUI. In addition, some improvements have been made to AGXUnity's DeformableTerrainParticleRenderer to accommodate an increasing number of particles in the visible model. The physical parameters of soil that can be set in the TerrainMaterial within the DeformableTerrain component are diverse. AGXUnity provides three preset types. The AGX version simulator selects the dirt_1 parameter preset by default. It is recommended for simulating typical viscous mud [15] like our test site.

4.3 Physical Interaction

In the AGX version, it is possible to calculate the interaction force between construction machinery and soil ground when the ground is collapsing or deforming, which was not implemented in the PhysX version. This is achieved through a two-step process involving the DeformableTerrainShovel from AGXUnity. First, this component is attached to the bucket link. Then, several key parameters are set: the Top Edge, Cutting Edge, and Cutting Direction. Finally, the bucket link is registered in the Shovels list within the DeformableTerrain component, as previous section.

5. Conclusion

This paper introduced two types of dynamic simulators provided by the open platform for Autonomous Construction OPERA. The OPERA platform offers two simulators: one based on the PhysX engine, available at no cost except for the Unity

license, and the other using the AGX engine. While the PhysX version is freely accessible, it exhibits comparatively less refined soil deformation behavior in interactions with construction machinery than the AGX version. Conversely, the AGX version requires a paid license for its dynamics engine. Both versions are published on GitHub, making them accessible for any users. We highly welcome user feedback as it is instrumental in our ongoing process to enhance OPERA's future convenience and functionality.

References

- [1] Genki, Y., et al: "Proposal of an Open Platform for Autonomous Construction Machinery Development", Proceedings of the 40th ISARC, 2023.
- [2] GitHub: pwri-opera. On-line: <https://github.com/pwri-opera>, Accessed: 26/12/2023.
- [3] Open Robotics: ROS. On-line: <https://www.ros.org>, Accessed: 26/12/2023
- [4] Open Robotics: GAZEBO. On-line: <https://gazebo.org>, Accessed: 26/12/2023
- [5] OROCOS. On-line: <https://orocos.org>, Accessed: 11/3/2024
- [6] OpenRTM-aist. On-line: <http://www.openrtm.org>, Accessed: 11/3/2024
- [7] Unity Technologies: Unity. On-line: <https://unity.com>, Accessed: 26/12/2023
- [8] Epic Games, Inc.: Unreal Engine. On-line: <https://www.unrealengine.com>, Accessed: 26/12/2023
- [9] Russel Smith: Open Dynamics Engine. On-line: <http://ode.org>, Accessed: 26/12/2023
- [10] Bullet Real-Time Physics Simulation. On-line: <https://pybullet.org/wordpress>, Accessed: 26/12/2023
- [11] NVIDIA.DEVELOPER: PhysX. On-line: <https://developer.nvidia.com/physx-sdk>, Accessed: 26/12/2023
- [12] Algorix Simulation AB: AGX Dynamics: Real-time multi-body simulation. On-line: <https://www.algorix.se/agx-dynamics/>, Accessed: 26/12/2023
- [13] Daniel, H., et al: "Real-Time Mud Simulation for Virtual Environments", ACM Siggraph Symposium on Interactive 3D Graphics and Games, i3D, 2018.
- [14] Algorix: AGXUnity. On-line: <https://github.com/Algorix/AGXUnity>, Accessed: 26/12/2023
- [15] Tomas, B., Mertin, S., Tech report V1.01agxTerrain, https://www.algorix.se/download/agxTerrain_tech_report.pdf, Accessed: 26/12/2023