Dedicated simulator for e-training of robot "Dromader"

I.Ostrowski, A.Masłowski
Research and Academic Computer Network (NASK),
Kolska Str. 12 01-045 Warsaw, Poland

E-mail: iostrowski@wp.pl

Abstract. Project of robot in simulation system can be divided into two main parts. First is graphical model, and second is physical model. This paper discusses results of using tools for 3D modelling and building physical models. Graphical models are developed using 3dsMax software from CAD model. It is possible to perform straight conversion from the CAD model to a graphical game-ready model but results are poor. The most serious issues were performance, and texturing. Graphical models had to be created from scratch. Mechanical part was developed with Vortex Editor. Vortex Editor is tool which allows creating mechanical system for Vortex physics universe. It is CAD like software, which allows creating project, running test simulation, and tuning parameters without exporting model to simulation framework.

1. Introduction

This paper describes simulator of UGV Dromader. build for training purposes. This UGV is very hard to operate and need long time of training for operators. Our simulator is shortening time of use robot for training purposes.

2. UGV Dromader

UGV Dromader robot (Figure 1) was designed and built by Military University of Technology, Warsaw. Robot's structure is characterized by a segmented chassis connected using an articulated joint. The turning angle of the front segment in relation to the rear can reach up to 90°. This provides high maneuverability, allowing for easy turns in confined spaces. In conjunction with the track drive system and both axis drive it allowed the "Dromader" to achieve a very good off-road mobility.

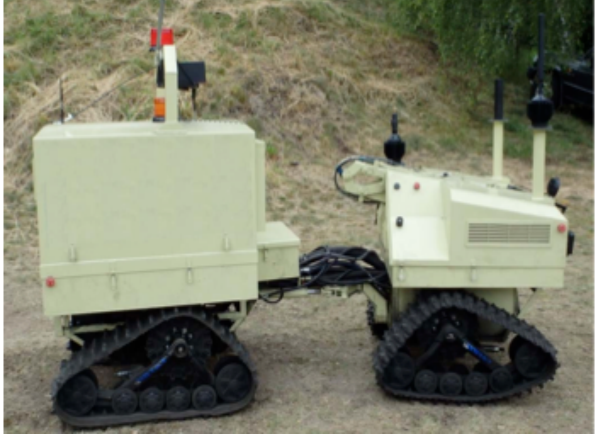


Figure 1 Robot Dromader

The rear section houses a 15kW combustion drive unit while the front section houses the control system, and several components of the platform's drive system. Because of the desire to minimize the weight of the robot, the entire chassis of the robot is made of aluminium.

The robot may be equipped with an experimental protection system against remotely detonated IEDs, which may be mounted in the rear section above the engine inside the chassis. Using this configuration requires the detector and jammer antennas to be mounted in very specific locations to avoid interference and not to reduce their working area due to chassis structure and other equipment mounted on the robot. The rear segment also contains a Wi-Fi transceiver antenna (2.4 GHz) for the remote-control system.

The manipulator had been mounted on the front section of the robot. Part of its structure and location requirements were that in its transport position it would not obscure the operator's field of view registered by the teleoperation system.

If you don't wish to use the Word template provided, please set the margins of your Word document as follows. The manipulator has three degrees of freedom allowing for control of the effector's position in the vertical plane. Due to weight limitations, the horizontal plane control is achieved through turning of the entire front section instead of an additional control mechanism.

3. Dromader simulator

3.1. Simulation framework description

Currently developed simulation system is a universal tool for multi-robot operation. Created software solution can be divided into two sub-projects. First is simulation server which is standalone binary package. This program runs a simulation server; this process is responsible for physics solving, multiple-client scene synchronization and watching mission progress. Second is a library which allows communicating with the simulation server. This software is dynamic linked library. This allows separating simulation and trainees' consoles, so simulation process can be run on multiple machines robustly. Another important advantage of presented solution is fact, that rendering scene with multiple cameras can be heavy task for computer, but it no longer affects physics simulation. Scene is synchronized via mixed TCP and UDP protocol. It allows connecting up to ten clients.

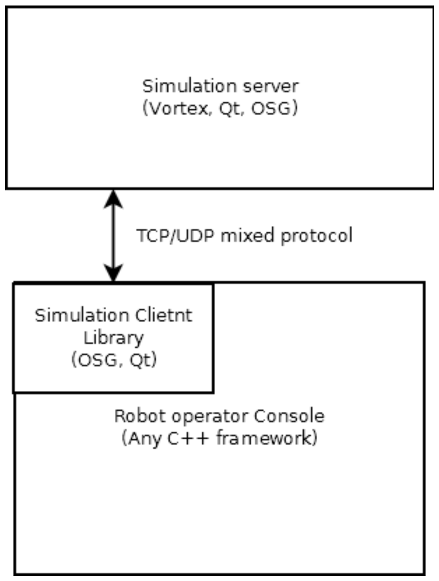


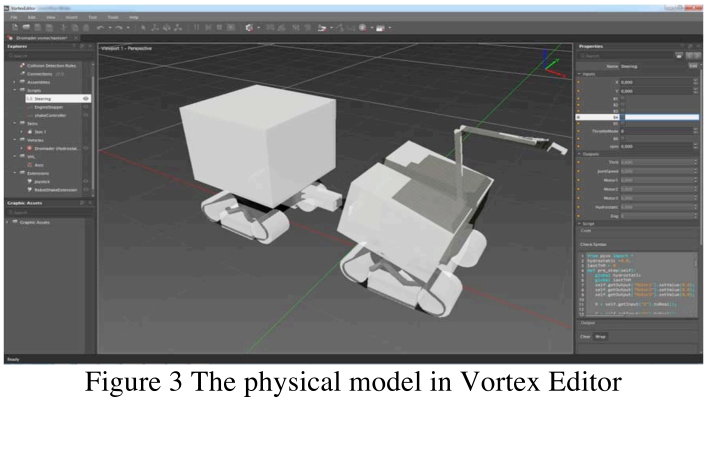
Figure 2 Simulation system concept

For physics simulation CM-Labs Vortex library is used. This library is solution for serious games, industry simulation (like cranes, excavators) and military training. It comes with a mature framework which allows developing simulation of almost any kind of vehicle. Framework allows simulating internal combustions engines, electric motor, variety of transmission systems including classical manual and hydrostatics transmissions, and nearly any kinematic layout. Also it is possible to simulation complex structure like robotics manipulators. Moreover some situation in simulation can be scripted using Python language. Library is proprietary and closed source, it is well integrated with Open Scene Graph.

OSG, another library, is open source, OpenGL based graphics engine. It is C++ library which is very often chosen in serious games and training system. This library provide number of useful tools like: a hierarchical structure of the scene, a hierarchical structure of graphical model with number types of relationship between model parts, graphical models loaders including proprietary formats like FLT, Collada or FBX, levels of details. It also supports OpenGL implementation from 1.1 up to 4.0 including OpenGL ES. Next part of simulation framework is Qt 5 library. This popular LGPL licensed open source C++ framework. It is widely used for GUI application, but it provides plenty solutions for non GUI developers. In project QtXML and QtNetwork modules were heavily used. Last used library is Boost.Python. This library is part of Boost open source set of C++ libraries. Boost.Python allows exposing some dynamically allocated C++ objects in Python name-space. This allows creating tools which can be used during execution of simulated mission scenario. For example, during mission scenario development, some triggers can be prepared. Those triggers can cause number of actions, like adding penalty points, time measurement and so on. Because of Python language, compilation is not needed. Code is loaded and interpreted by integrated Python parser, in runtime.
Simulation client used only OSG and Qt libraries. It makes this solution GUI – ready, multi-platform and lightweight. It can be easily integrated with operator or trainer console. This library deals with recreating graphical scene in client software, rendering camera views, and sending and receiving data from and to robots. It also has methods to receive some messages from simulation server, which can be sent from scenario's script executed on server's side.

3.2. Dromader robot simulation

Project of robot in simulation system can be divided into two main parts. First is graphical model, and second is physical model. It is important that second part is based on created graphical model. Graphical models are developed using 3dsMax software from CAD model. It is possible to perform straight conversion from the CAD model to a graphical game-ready model but results are poor. The most serious issues were performance, and texturing. Graphical models had to be created from scratch. Mechanical part was developed with Vortex Editor. Vortex Editor is tool which allows creating mechanical system for Vortex physics universe. It is CAD like software, which allows creating project, running test simulation, and tuning parameters without exporting model to simulation framework.



```
from pyvx import *
Hydrostatic =0.0
lastTHR = 0
def pre_step(self):
    global Hydrostatic
    global lastTHR
    self.getOutput("Motor1").setValue(0.0)
    self.getOutput("Motor2").setValue(0.0)
    self.getOutput("Motor3").setValue(0.0)

    X = self.getInput("X").toReal()
```

Figure 3 The physical model in Vortex Editor

The graphical model is organized in hierarchical structure, where every moveable part has own node, with own coordinate system. During physics modelling number of properties were added. First of all collision geometries were created. Collision geometries are simple primitives like boxes, cylinders that interact with each other's. Collisions are not resolved between graphical meshes. These shapes are organized in to parts. Every collision shape has own mass, inertia matrix and material. Afterwards, constraints were created. These are objects, which create connection between two or more parts. It could be for example prismatic, or hinge pair. A number of parameters describe constraint, like character (free, motorized or locked), internal friction, damping and stiffness, motor or blockage maximum force, and limits. In this fashion hydraulics actuators were added to the simulation. Next robot-vehicle system was added to physics model. There are some typical vehicle template like cars (AWD, RWD, FWD with auto and manual transmission), tracked vehicles with skid-steering like excavator (hydrostatics torque division) or tank (differential torque division). Robot Dromader is a unique vehicle solution so new template was created.

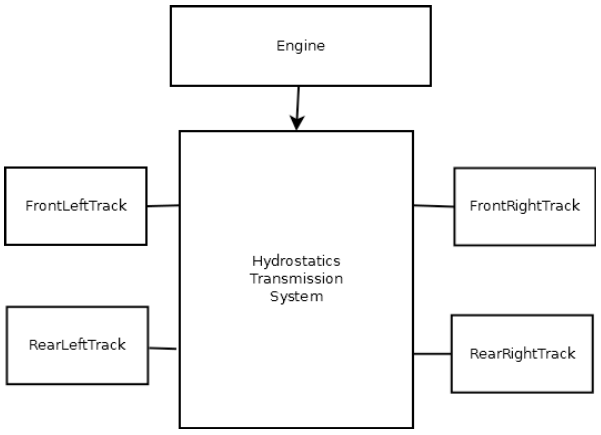


Figure 4 Robot Dromader system template

Template, after loading to Vortex Editor automatically creates such complex structures like AWD truck. Objects like engine, differentials, shafts, transmission system, driving constraints and finally wheels with suspension are introduced, and put in proper relationships. Structure of Dromader robot is pretty simple. The engine is connected to the hydrostatics transmission, and ratio of every output motor can be set separately. The robot doesn't have differential steering (like excavator), so ratio for each track is the same. Driving is realized using hinge constraint with velocity actuator. Final layer of physics model is logic, which does not have to be hard coded in application code. Every robot's object can communicate with Python scripts. These scripts are an elegant way to implement some logics. Example – Dromader robot has three driving modes. Switching it affects motor velocity, but also reconfigures hydrostatics system, connecting motors parallel, thus higher torque is generated. This function cannot be implemented directly, but can be realized as script which would change hydrostatics system ratio in proper way.

Last, but not least is VHL interface. This tool allows creating a universal interface to communicate vehicle model with rest of application. After loading model in final simulation application it allows to write and read parameters from C++ code.

3.3. Environment

Sample environment was created from point cloud. It's much easier to create real location model using geodetic data than only photos. Environment also was created using 3dsMax. For buildings floor plan was extracted from point cloud data, for ground shape an elevation map was created. Vegetation was replaced with simple billboard representation. Billboard is a graphics method for creating planar shapes that always face viewer's camera. This method is widely used for simple vegetation rendering

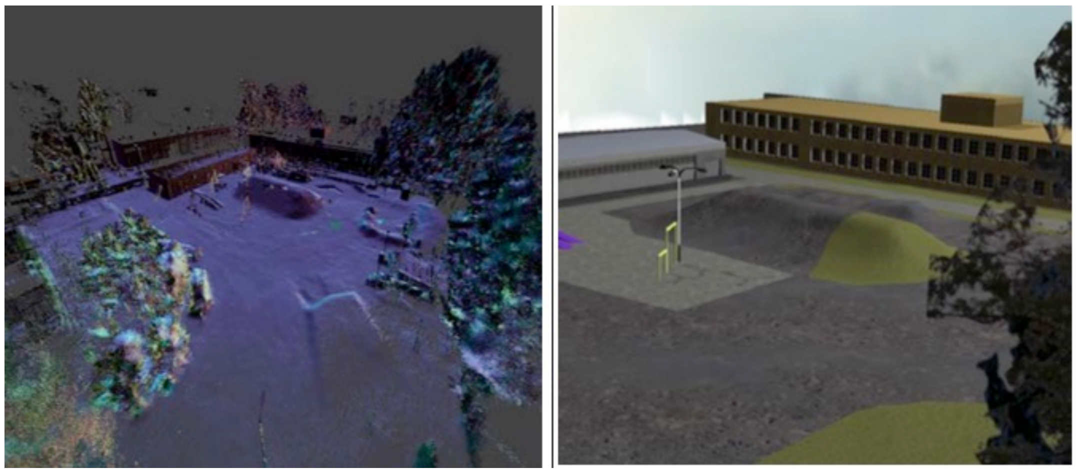


Figure 5 Environment representation - left point cloud, right 3d model in Vortex simulator

Scene was textured, and finally imported to Vortex editor where materials parameters were applied to surface.

Configuration is object orientated XML file which simulation server loads during initialization. These files contain a great deal of information, like spawn position of robots, scene type, position 3rd person view cameras, and so on. It allows to set-up, or modify simulation scene in very convenient way. What is more, XML file contains a scripts coded in Python language, that allows to interact with simulation, so some events can be dynamically created. It can create a lot of opportunities to develop live- like, interactive scenarios.

3.4. Operator console

Operator console, as was mentioned before, had been created using Simulation Client Library. The Dromader operator's console is hand-held computer with game-style controller. It is communicate with robot using wireless connection. Operator can observe one camera at the moment, and steering robot and its manipulator using analog sticks and buttons on the controller. Operations like connection, disconnection, switching on and off robot’s system are realized via touch screen GUI. Due to lack of access to source code of operator's console, interface was recreated using own framework. This program was successfully integrated with Simulation Client Library, and finally real-like operator's console was introduced. Separation of Simulation Client Library and console’s code is great advantage, because it makes simulator independent from used input and output devices and allows performing integration with real equipment.

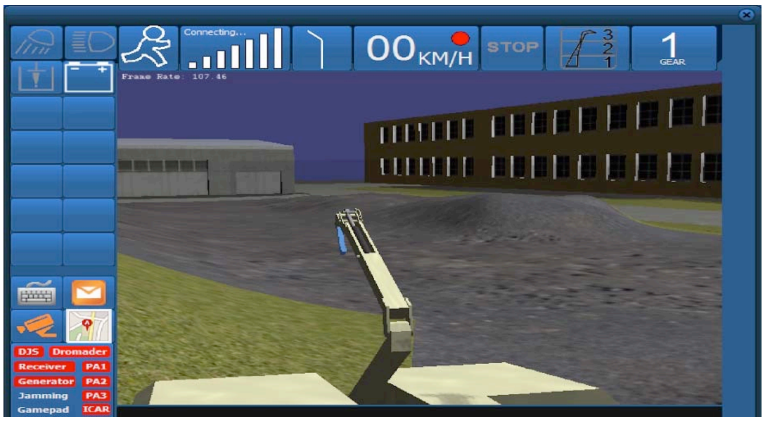


Figure 6 operators console connected to simulator server

4. References

[1] Maslowski, A. (2014) Training in Military Robotics and EOD Unmanned Systems. Conference and Seminars, 30 September - 2 October 2014, NATO EOD Center of Excellence, Trenčín, Slovakia.

[2] Łuczyński T., Ostrowski I., Będkowski J., Masłowski A., Przybyłko M., Zoppi M., Wojcieszyńska P., Szczepaniak M., (2013) Mobile Demining Devices Modeling for the Training Purpose using VORTEX Framework, Proc. of the Int. Conf. on Military Engineering – Problems and Prospective, 23-25 April, 2013, Institute of Military Engineering, Wroclaw, Poland, pp. 233-240.

[3] Kaczmarczyk, A., Kacprzak, M., Masłowski, A. (2009a) Muti-level simulation training of devices operators. An application to mobile robots operators (in Polish). Elektronika 11/2009.

[4] Bedkowski J., Majek K., Ostrowski I., Musialik P., Maslowski A., Adamek, A., Coelho, A., De Cubber, G., “Methodology of training and support for urban search and rescue with robots” ICAS 2013, “The Ninth International Conference on Autonomic and Autonomous Systems”, Lisbon 2013, Portugal

[5] Ostrowski, Igor, et al. "Training tools for TIRAMISU Project." Annual Report fp7-tiramisu.eu

[6] Musialik, P., Będkowski, J., Ostrowski, I., Łuczyński, T., Majek, K., Masłowski, A., & Adamek, A. (2015). Virtual environment modeling for training of unmanned vehicles’ operators. Zeszyty Naukowe/Wyższa Szkoła Oficerska Sił Powietrznych.