Proceedings of the International Conference on Industrial Engineering and Operations Management
Paris, France, July 26-27, 2018

PackageCargo - Open Source Tool based on Optimization and Simulation for the Container Loading Problem with Dynamic Stability

Juan C. Martínez, Daniel Cuellar, Edgar Céspedes and David Álvarez-Martínez
Department of Industrial Engineering
School of Engineering
Universidad de Los Andes
Bogotá D.C., Colombia
d.alvarezm@uniandes.edu.co

Abstract

Traditional considerations related to dynamic stability constraint in the Container Loading Problem (CLP), such as boxes with insufficient lateral support, have proven to be inaccurate when compared with the results of dynamic simulations where the boxes are subjected to real world conditions. In response to this, two indicators for dynamic stability were introduced; the number of fallen boxes and the number of boxes likely to be damaged in case of acceleration. These indicators, however, have not yet been estimated using a mechanical approach. An open source application (PackageCargo) was developed to calculate, visualize, and save efficient packing patterns to instances of the CLP, and has been enhanced with a dynamic simulation environment to obtain performance indicators related the dynamic stability of such patterns. The physics engine used in the application is capable of trading accuracy for simulation speed and is in most cases non-deterministic. Through PackageCargo we evaluated the accuracy of new dynamic stability metrics. A mechanical model that measures dynamic stability without the need to subject the packing pattern to a physics simulation was developed, based on a dynamic analysis of the forces and accelerations acting upon the boxes and using the kinetic parameters of the load such as mass distribution, coefficient of friction, and rigidity.

Keywords
Container Loading Problem, Decision Support Systems, Dynamic Stability

1. Introduction

The Container Loading Problem (CLP) is one of the most relevant among cutting and packing problems. The CLP has been proven to be NP-Hard (Pisinger, 2002), which in addition to its large spectrum of applicability, has allowed researchers and industry alike to maintain interest on the problem. The CLP seeks to pack a set of small rectangular items into a single rectangular container of bigger size, aiming to maximize the utilization of space within the container.

Real CLP applications require the modelling and consideration of practical constraints of significant complexity. The authors in (Bortfeldt & Wascher, 2013) present a compilation of constraints related to the container loading problem. Among several types of constraints, this work focuses on the cargo stability. The cargo stability constraint seeks to preserve the integrity of the items (customer satisfaction) and the safety of operators carrying out loading/unloading operations. Previous works that address cargo stability have defined two types of stability: static and dynamic. Static stability refers to the equilibrium of the box (maintaining the position of the cargo) during loading and unloading operations, whereas dynamic stability refers to the equilibrium during transportation and is the least studied type of stability in literature. (Ramos et al., 2015) proposed a set of dynamic stability metrics for the CLP, which reflect dynamic stability based on the crucial factors of integrity and safety. The contributions of this work arise from these metrics.

In this work, dynamic stability metrics found in the literature were reviewed and reformulated, since some of these did not contemplate the total number of boxes found in the packing pattern, which would not allow a

comparison to be made in future works that wish to solve this problem. This is followed by the development of an open source simulation and optimization tool in Unity®, which allows to simulate the behaviour of a packing pattern when applying the most common accelerations encountered when transporting cargo on land roads. For this, a series of interfaces were developed to allow the communication between the tool and different schemes of simulation and optimization. Currently, the simulation can be run by either the physics engine PhysX® (included in Unity) or an *ad hoc* simulator, like a mechanical model that can estimate the values of dynamic stability indicators without running a simulation tool, with the objective of having the model coupled to an optimization scheme in future works. The optimization (packing pattern generator) can be performed by any algorithm linked to PackageCargo through shared-memory or system files. In this work we used a GRASP Algorithm proposed by (Álvarez-Martínez et al., 2015) for the CLP considering box orientation, load bearing strength, and static stability (full support) constraints; considerations needed when the dynamic stability constraint is considered. Finally, an information storage module that registers relevant information for future work has been developed.

The tool proposed (PackageCargo, see Figure 1) in this work allows researchers and the public to integrate their developments, whether they are oriented towards simulation or optimization without the need to implement an entire framework. In this way, works focused on simulations (by default) are coupled to an optimization algorithm that allows to find efficient packing patterns. In the same way, works focused on optimization can be easily coupled by shared-memory (to maintain their computational performance) to the simulation module that runs in fractions of a second.

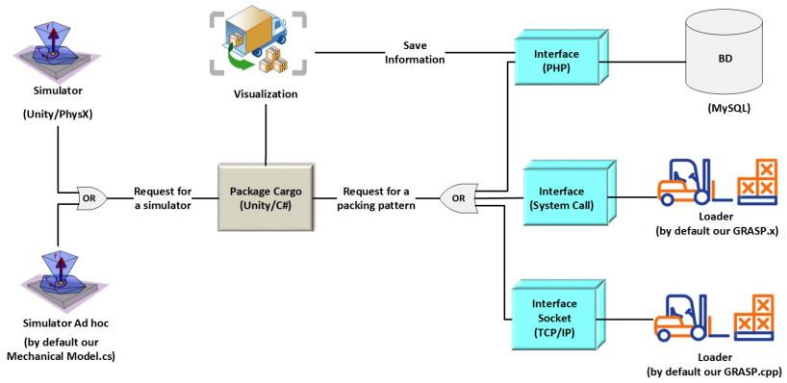


Figure 1. Architecture of PackageCargo

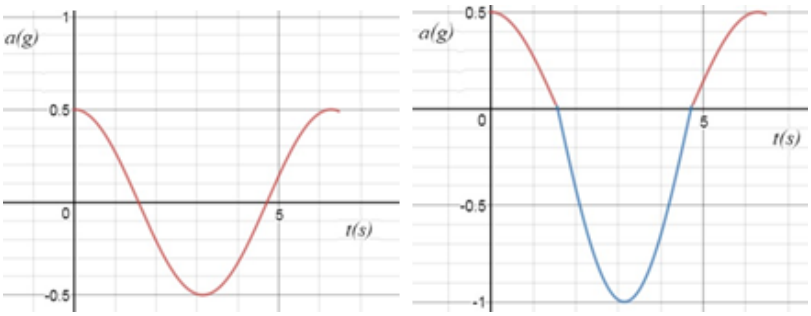
2. State of the art

2.1 Dynamic Stability Metrics

Dynamic stability metrics predict how much the geometric configuration of a loading pattern will change during transport. M1 and M2, the traditional performance indicators for Dynamic stability (M1 and M2) do not require dynamic simulations to be performed, while two novel metrics, NFB and *NB DBC*, proposed by (Ramos et al., 2015) do. The different metrics are defined as follows:

- *Number of boxes supporting load (M1):* It is the average number of pieces supported by each box that is not located in the base of the container. This metric gives an estimate of how intertwined the boxes are; the higher its value, the greater the dynamic stability of the loading pattern. Its readily obtainable from the geometry description of a loading pattern.
- *Percentage of boxes without sufficient lateral support (M2):* The ratio of the number of boxes without support against a wall or another box on at least three of their four lateral faces, to the total number of boxes. It correlates inversely with dynamic stability and can also be calculated without resorting to a dynamic simulation.
- *Number of fallen boxes (NFB):* It is the number of boxes that present a difference in position along the vertical axis of the container between the initial and the final states of container movement. Unlike M1 and M2, this metric is typically not obtained from analysis of a stationary state, but rather from a dynamic evaluation of the configuration. In simulation, only the initial and final states are of concern for the calculation of NFB. It is clear that the higher this number is, the more unstable the loading pattern becomes.

- *Number of boxes within the damaged boundary curve (NB DBC):* The number of boxes that suffer either a drastic change in velocity (sliding) or an excessive acceleration for a given moment (bumping), that might cause considerable damage to the box or its contents. The parameters of maximum velocity delta and maximum instantaneous acceleration are obtained from standard fragility tests as described in ASTM D3332-99(2010). This metric is more computationally expensive than NFB, as it requires careful tracking of the velocity of each box, not only through continuous acceleration, but also during discrete impact events. Its inversely correlated related to dynamic stability.



(a) Sideways acceleration (b) Forward acceleration

Figure 2. Acceleration functions

The new metrics were modified to take the total number of boxes of the packing pattern into account, all in order to facilitate comparisons between arrangements of different number of packed boxes for the same instance. The modified metrics are M3 = NFB/#*packedBoxes* (percentage of fallen boxes divided by the total number of packed boxes) and M4 = NB DBC/#*packedBoxes* (percentage of boxes within the damage boundary curve divided by the total number of packed boxes). The dynamic simulation environment subjects the load configuration to the accelerations most frequently experienced by cargo transport vehicles, shown in Table 1., as defined by the Container Handbook (Mitchell, 1998).

The British regulations, which provide the highest values for Forward and Lateral accelerations, were chosen as a reference point for the simulation parameters. To prevent unwanted effects caused by jerk, the derivative of acceleration in time, the movement of the container is defined by sine functions, as described in Figure 2a for lateral accelerations.

Table 1. Acceleration according to regulations (GDV, 2017).

Norm	Forward Acceleration	Braking Acceleration	Lateral Acceleration
VDI	0.8g	0.5g	0.5g
CTU	1.0g	0.5g	0.5g
Swiss regulations	1.0g	0.5g	0.5g
British regulations	1.2g	0.5g	0.8g

Forward and braking accelerations are described by a part function, shown in Figure 2b, allowing the two accelerations to be obtained in one continuous movement. The mentioned accelerations must be provided as inputs into a physics engine to run the dynamic simulation from which NFB and NB DBC are obtained.

2.2 Physical Motors

Physics Engine: is a piece of software that simulates physical phenomena. They range in accuracy form high precision, deterministic software used in aircraft design and stress analysis to stochastic middleware intended for real-time simulation. Regarding the study of dynamic stability in the CLP, a physics engine specialized in rigid body dynamics and Newtonian mechanics is of most interest. These two concerns are usually the least computationally

expensive physical systems to simulate, and as such, even real-time physics engines can run them with enough accuracy (Boeing & Brunl, 2007).

Stacked objects Problem: in a packing pattern, the sides of boxes are never perfectly flat, nor is the centre of mass always in the geometric center, Therefore, it is exceedingly difficult to accurately predict the behaviour of stacked objects. Further difficulties arise from small variations in materials that affect friction models, as well as a plethora of other effects that alter the way force is transmitted between stacked objects.

However, when noise is added in deterministic solvers, and in most iterations of non-deterministic ones, most physics engines can simulate stacked objects in a verisimilar, if not a mathematically exact, manner (Boeing & Brunl, 2007).

3. Methodology

3.1 PackageCargo and Unity Assesment

PackageCargo: is an open source application developed to calculate, visualize, and save efficient packing patterns to instances of the CLP (see Figure 3). The interface was based on the commercially available tool (EasyCargo, 2017), while the simulation environment was inspired by StableCargo (Ramos et al., 2015), a tool that measures NFB and NB DBC. The following design parameters were considered in the construction of the application:

- Cargo arrangement visualization: Existing and new loading patterns are to be visualized in a three-dimensional representation, with a changing view point and occlusion (hiding) of packages.
- Cargo definition: It must be possible for the user to define new load units (packages) as well as container sizes; and to save them into a database for other users.
- Simulation: Through definition of rigid bodies and colliders (Erleben, 2002) for each box, a dynamic simulation environment must be implemented, where applied forces modify the velocity and position of the loading pattern according to Newtonian mechanics, with the assumption of constant density for every box. The results of the simulations must be visualized as well.
- Performance analysis: The application must include a way to calculate and/or report the performance indicators of the loading patterns, as well as save them to the database.
- Connectivity: Besides a centralized database for the former, PackageCargo should be distributed as both a standalone program and a web client to run on less capable hardware.
- Multiplatform: To reach as many users as possible, regardless of the web client, the application should be able to run natively on a multitude of operating systems and server.

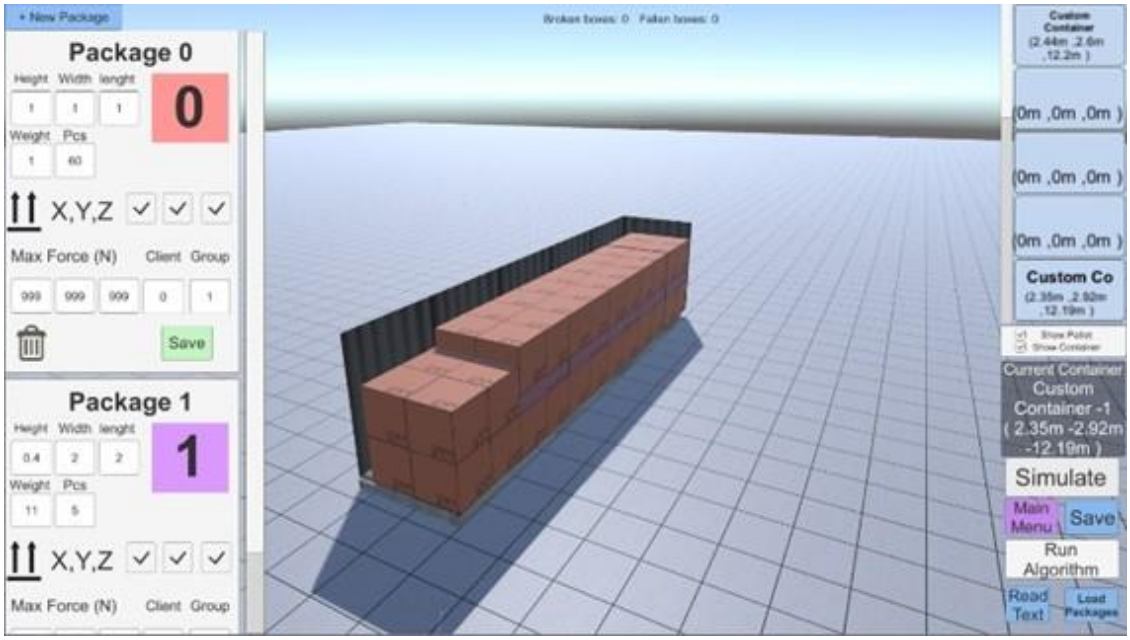


Figure 3. PackageCargo interface

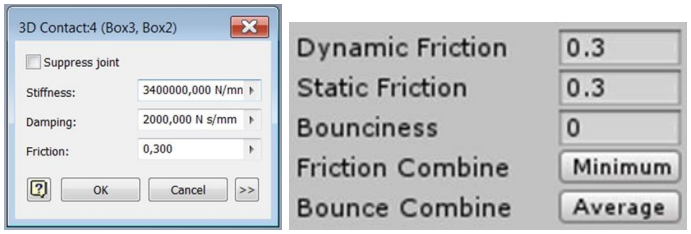
Coding an application that meets the design criteria, particularly the rendering portion, would take considerably more resources than a development suit. To accommodate for these requirements a game engine, Unity, was selected as the main tool for creating PackageCargo.
(Unity Techologies, 2017) is a development platform for interactive software, particularly videogames, designed to publish applications to multiple platforms. As a game developing suite, it incorporates many subsystems that are of use at meeting the design parameters. It can use DirectX ®, Vulkan ® and OpenGL ® as rendering APIs, has a rich programming environment based on C, and as such can use the system utilities associated to it, including TCP/IP connectivity and file system access, among others. Unity can export builds to 32 and 64-bit versions of Windows ®, Linux ® and macOS ®; WebGL applets, and even ARM apps for Android ® and iOS ®.

The application was built on Unity, and the physics engine used by default in the simulation environment is PhysX SDK 3.0. PhysX (NVIDIA Corporation, 2016) is a physics engine middleware, frequently used mostly in the real-time simulation of physics, and is frequently implemented in videogames. This means that the engine is capable of trading accuracy for simulation speed by forcing calculations to stop if the time consumed exceeds a maximum timestep. This marks one of the biggest weaknesses of PhysX, it is in most cases nondeterministic. Other limitations, such as lack of soft body simulation and Coriolis accelerations are not relevant when simulating to obtain dynamic stability metrics.

Autodesk Inventor® (Autodesk Inc., 2016) is a parametric modelling program frequently used in computer assisted mechanical design. It contains a dynamic simulation environment with a solver based on Runge-Kutta integration capable of evaluating a mechanical systems behaviour when affected by conditions of force, torque, mechanical joints and imposed motion. Inventor's friction model is based on the definition of a single friction coefficient that modifies a contact force function that has as parameters the normal force and the relative velocity of the contact surfaces. This contrasts with PhysX's simpler kinetic and dynamic friction separation.

3.2 Accuracy of PhysX, through a benchmark test

A simple benchmark test comparing PhysX with software designed for accuracy in dynamic simulations (Autodesk Inventor) was performed to evaluate the accuracy of PhysX. The benchmark test consisted in stacking three identical boxes within both Inventor and Unity, and then applying an acceleration of 10 m/s^2 to the middle box (coloured red), like a tablecloth trick. Figure 4 shows the friction and contact parameters used in the tested and the referenced simulator.

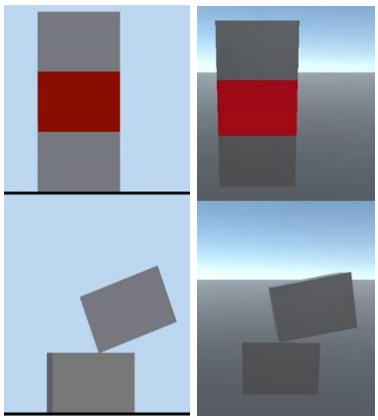


(a) Autodesk Inventor (b) Unity

Figure 4. Parameters used in the simulation benchmark

The systems evolution from 0 to 3 seconds was evaluated. Inventor completed the simulation in 3 minutes and 40 seconds, while Unity performed in a timescale of 0.5 with respect to real time, that is, it was locked to be completed in 6 seconds.

As Figure 5 suggests, results were similar enough to indicate that the developed application has an adequate precision, while producing the results in a fraction of the time used by the standard simulation software.



(a) Inventor (b) Unity 5.6

Figure 5. Starting and final states for the simulation

3.3 PackageCargo open source tool

PackageCargo is not just a container loading visualization and dynamic simulation software. Figures 1 and 3 illustrate the architecture and interface followed by the program, there the connection between the information storage and optimization modules is made evident. The architecture consists of four components; the visualization module, composed of a geometry renderer and a user interface that includes indicators and simulation and optimization results; a simulation module, that tests the stability of the loading pattern when subjected to motion; an optimization module, that allows the inclusion of different algorithms for the generation of loading patterns; and a database component that manages registered data and data generated by the optimization and simulation modules.

The optimization module has been associated to a GRASP algorithm included by default, nevertheless it is possible to use other loading algorithms by a socket connected to a server or by directly calling executable programs so that users can compare them and generate reports on the performance indicators relating to stability. For the streaming of data through the socket, a connection by TCP/IP is established in a client-server architecture, meaning that external loading algorithms must be defined as servers that receive requests from the client application (PackageCargo). In the case of program executables (standalone architecture), the user must build their algorithm as an executable supported by their target platform (Windows, macOS or Linux).

The simulation module can choose between a simulation routine using Unity's physics engine PhysX SDK, or an ad-hoc mechanical model. A model not presented in this work (Mechanical Model.cs) is enabled by default. Given a packing pattern, this module determines the consequences of subjecting the boxes to the most critical conditions that could be expected from the transportation of the container, namely acceleration and braking, and left and right turning. The simulation module will also produce the dynamic stability indicators reviewed in this work. This is done with the purpose of evaluating loading patterns or to include the indicators in an optimization scheme.

The data storage module manages all information, whether it be registered by the user or generated by the other modules. This is registered on a database (MySQL®) located on a centralized server, which any user can access by request. This module performs calls to register, update and delete information related to user-created instances. It has a PHP interface that modifies the database by SQL commands. The database stores package characteristics (dimension, destination, load bearing strength, etc.), containers and loading patterns (geometry and performance indicators).

4. Experimental results and analysis

The PackageCargo was coded in Unity (C#). All the experiments were performed on a Windows machine of 64 bits with a processor Intel Core i7-7700HQ, 16 GB of RAM and a GPU NVIDIA GeForge GTX 1050 Ti 4GB. To carry out all the benchmark tests were used the classical set of instances presented in (Bischoff & Ratcliff, 1995). The 1500 problems are divided into 15 classes (BR1-BR15) of 100 instances.

The basic dynamic stability indicators (Bischoff & Ratcliff, 1995) (columns M1 and M2 from the Table 2) were calculated as the average value of the measured for each packing pattern produced by the GRASP algorithm.

To measure the M3 indicator for each instance it was necessary to report the mean value corresponding to each class. Due to non-deterministic nature of the physics engine, every instance was run 30 times, reporting the average (column M3(Ph)) for each class.

On the other hand, PhysX allows to measure M4 for each instance. For this indicator it is necessary to include the fragility of the boxes. For this study we have used two values that according to (Mitchell, 1998), correspond to the interval of maximum values of acceleration (3g-20g) that products classified as "fragile" may withstand. For future comparisons M4 was calculated using the extreme values of acceleration (3g and 20g), the mean value was reported running 30 iterations per instance for each class (columns M4(3g) and M4(20g), respectively).

The results obtained from PhysX follow the trend of lower M3 and M4 values for higher percentages of volume utilization, and, as suggested by (Ramos et al., 2015), the dynamic stability metrics show a much weaker relation with M1 and M2. When the items are defined to have a lower fragility or maximum allowable acceleration (20g vs 3g), M4 decreases drastically. M3 however, doesn't vary appreciably; after all the fragility of an item has no effect on its kinematic properties.

The values for M3 calculated follow the trend of higher values for better utilization of space within the container.

Table 2: Aggregated performance indicators for the benchmark instances (Bischoff & Ratcliff, 1995)

Class	**M1**	**M2**	**M3(Ph)**	**M3(PM)**	**M4(3g)**	**M4(20g)**
BR1	1,78	12,67	2,68	2,36	9,02	0,03
BR2	1,73	14,20	2,75	2,44	8,81	0,05
BR3	1,52	17,43	2,03	1,80	6,87	0,03
BR4	1,52	18,06	1,81	1,85	6,52	0,03
BR5	1,53	20,47	2,19	2,25	3,80	0,03
BR6	1,47	22,33	2,39	2,52	4,01	0,03
BR7	1,42	25,63	2,84	3,04	4,86	0,03
BR8	1,35	31,41	2,90	3,05	6,91	0,02
BR9	1,28	36,54	3,33	3,85	7,14	0,03
BR10	1,24	38,73	4,30	4,12	14,65	0,02
BR11	1,22	41,72	4,87	4,90	15,13	0,02
BR12	1,19	42,67	5,57	5,49	15,49	0,02
BR13	1,16	45,74	6,83	6,75	16,51	0,01
BR14	1,13	48,94	7,68	7,69	17,53	0,02
BR15	1,10	51,28	8,30	8,52	18,33	0,02
Avg.	1,38	31,19	4,03	4,04	10,37	0,02

5. Conclusions

PhysX is accurate according to the performed benchmark test. Additionally, the results show acceptable repeatability for the dynamic simulations, allowing it to be taken as an accurate baseline for comparison with the mathematical model.

PhysX produces results for M3 and M4 with satisfactory accuracy. This is of significance, since PhysX produces such results in a short amount of time, without presenting any variance in its results. This makes it attractive for integration with an optimization algorithm considering the dynamic stability constraint.

The open source nature of PackageCargo makes the software readily accessible for contributions seeking to enhance its capabilities beyond the scope of this work (i.e. simulating marine and air transportation). It also makes the offered optimization and simulation tools available to the scientific and academic communities to allow the evaluation of simulation models and optimization algorithms.

References

Pisinger, D., *Heuristics for the container loading problem*, European journal of operational research. 141. (2). pp.382-392. (2002).

Bortfeldt, A., Wascher, G., *Constraints in container loading A state-of-the-art review*. European Journal of Operational Research. 229. 1. pp. 1-20. (2013).

Ramos, A., Oliveira, J., Goncalves, J., Lopes, M. *Dynamic stability metrics for the container loading problem*. Transportation Research Part C: Emerging Technologies. 60. pp.480-497. (2015).

Alvarez-Martínez, D., Alvarez-Valdes, D., and Parreño, F., *A GRASP algorithm for the container loading problem with multi-drop constraints*. Pesquisa Operacional, 35(1). pp. 1-24 (2015).

Mitchell, P., *Material and part handling in manufacturing*. Dearborn: Society of Manufacturing Engineers. (4th ed.). (1998).

GDV, Gesamtverband der Deutschen Versicherungswirtschaft., *Container Handbook*. Cargo loss prevention information from German marine insurers. Ch 2. Available Online http://www.containerhandbuch.de/chb_e/. Accesed (02/04/2017).

Boeing, A., Brunl, T., *Evaluation of real-time physics simulation systems*. Proceedings of the 5Th International Conference On Computer Graphics And Interactive Techniques In Australia And Southeast Asia - GRAPHITE '07. (2007).

EasyCargo., *EasyCargo is truck and container loading software*. Available Onlinehttp://www.easycargo3d.com/. Accesed(16/04/2017).

Erleben, K., *Module based Design for Rigid Body Simulators*. Technical report. University of Copenhagen. (2002).

Unity Technologies., *Unity es el software líder a nivel mundial en la industria de los juegos*. Available Online http://www.unity.com/. Accesed (02/02/2017).

NVIDIA Corporation., *PhysX is a scalable multi-platform game physics solution supporting a wide range of devices, from smartphones to high-end multi core CPUs*. Available Online https://developer.nvidia.com/physx-sdk. Accesed (20/12/2016).

Autodesk Inc., *Inventor Professional 3D CAD software offers an easy-to-use set of tools for 3D mechanical design, documentation, and product simulation*. Available Online http://www.autodesk.com/. Accesed (11/12/2016).

Bischoff, E.E., Ratcliff, M.S.W., *Issues in the development of approaches to container loading*. Omega, 23: pp.377-390. (1995).

Biographies

Juan Camilo Martínez received the Engineering and master's degree in mechanical engineering from Universidad de Los Andes, Bogotá, Colombia, in 2018.

Daniel Cuellar received the Engineering degree in Industrial Engineering from Universidad de La Salle- Bogotá, Colombia, in 2017. He is a master student in Industrial Engineering at the Universidad de Los Andes (Colombia) since 2017.

Edgar Céspedes received the Engineering degree in Industrial Engineering from Universidad Central - Bogotá, Colombia, in 2017. He is a master student in Industrial Engineering at the Universidad de Los Andes (Colombia) since 2017.

David Álvarez-Martínez received a PhD from Universidade Estadual Paulista "Júlio de Mesquita Filho" (Brazil) in 2014. He is a Professor of Industrial Engineering at Universidad de Los Andes (Colombia) since 2016. His current research interest includes optimization, simulation and industrial automation (robotics) for transportation and logistics problems.