

Simulation based Planning of Ferry Terminal Operations

Jayanta Majumder, Dracos Vassalos, Shikha Sarkar, Hyunseok Kim,

Department of Naval Architecture and Marine Engineering, Universities of Glasgow and Strathclyde,
{j.majumder, d.vassalos, s.sarkar, h.s.kim}@strath.ac.uk

Luis Guarin, Anthony York, Terje Dahlberg, Safety at Sea Ltd., Glasgow,
{l.guarin, a.york, t.dahlberg}@safety-at-sea.co.uk

Abstract

This paper presents the design of a software tool developed for modelling and simulation of the landside vehicular operations in passenger ferry terminals. The tool is an add-on module to the pre-existing crowd flow simulation tool called Helios [MAJUMDER, J.; VASSALOS D. et al. (2005)] and is named Helios-PortSim. This tool should be of interest to operations analysts and port facility designers seeking to evaluate the performance of a designed facility or operation with respect to vehicular throughput. This tool addresses the peculiarities of ferry loading/unloading operations that are impossible to model convincingly using the general purpose simulation tools such as discrete event simulators. This work was supported by a major European ferry operator and has been enriched with a wealth of practical knowledge and experience of the said operator. The tool has helped evaluating ferry loading/unloading operations for three busy ports and its results have contributed to some important design decisions regarding the acquisition and renovation of the ports' infrastructure. The paper discusses the requirements of Helios-PortSim and presents the software design that meets such requirements.

1. Introduction

The latest ferry ships are very fast and efficient on the waters but the typically slow port operations bring down the efficiency of the overall operation. Improving the efficiency of the port operations can make an appreciable difference in the revenue of ferry operators. In order to address this need we have investigated on and developed a software tool (named *Helios-PortSim* or simply *PortSim*) for planning of port operations through simulation.

This tool is specifically designed for planning of ferry-terminal operations. The general purpose discrete event simulators such as *QuestTM*, *FlexSimTM*, *AutomodTM* etc. are suitable for planning the operations of manufacturing, cargo handling, material handling, warehousing, packaging, etc., but when it comes to modelling the ferry terminal operations they fall short of some important modelling features. *PortSim*, on the other hand, has built in features for evaluating port-specific aspects such as placement and multiplicity of check-in devices, buffer area size, trailer yard layout, layout of parking areas for different types of vehicles, customs facility, vehicle exit facility, trailer operations, and so on. Three full scale case studies undertaken as a part of this development reveals that the trailer loading/unloading process, which is usually the longest sub-process of a ferry loading/unloading operation, can be improved significantly by proper planning of the operation.

In this paper, we present the design of the *PortSim* software and describe the modelling elements that are combined to create the wide variety of vehicular operations required in a ferry terminal. The modelling involves creation of a connected structural abstraction representing the port as a multi-layer traffic network comprising the parking lanes, ramps, shipboard lanes in multiple decks, connecting roads and nearby public roads. The behavioural model of the simulated vehicles includes goal-based and rules-constrained automatic planning of routes on the road network. A number of behavioural

elements such as manoeuvres, event triggers, mutual exclusion, queuing, deferred action etc. are combined in the task of modelling the simulation scenario. The simulation helps the operator to answer questions such as the following.

1. Can the requisite number of trailers be booked given a required turn-around time and a known number of tugmasters that can be deployed?
2. Can the same be done if the number of tugmasters is increased?
3. What is the maximum number of cars that can be loaded within a given time?
4. Given the arrival rates of an exceptionally busy period, will the incoming traffic spill over to the main road? How many additional check-in personnel (in addition to the usual self check-in devices) would be required to prevent the spill-over?
5. What would be the impact of a broken-down check-in device?

Visualization of the simulation is done using 3D interactive computer graphics. The state of the simulation can be altered dynamically during the simulation using mouse-clicks and keystrokes. The following sections present the requirements and the design of the PortSim system

2. Operations in a Ferry Terminal

The complete port operation consists of a complex interaction of several cooperating players including the ship operators' staff, the port management, passengers, tug-master operators etc. It is not necessary to model every single aspect of the operations. Only the operational aspects that contribute significantly to the ships' turn-around time and the ship operators' port usage time are of interest. With that limit on the scope there still are quite a few operational processes to model. Following is a list of those processes.

1. **ARRIVAL OF PASSENGERS:** Passengers start arriving a few hours before the vessel's departure time. An exponential distribution of inter-arrival time is normally assumed for steady state arrival processes, but that does not quite hold for passenger arrivals. The arrival processes for the ferry port has its own peculiarity. This is so because a large section of the passengers are aware of the ferry departure schedule, and typically want to arrive at the port as late as possible and yet make it on time for the loading. The last hour before the departure is extremely busy when cars arrive very frequently.

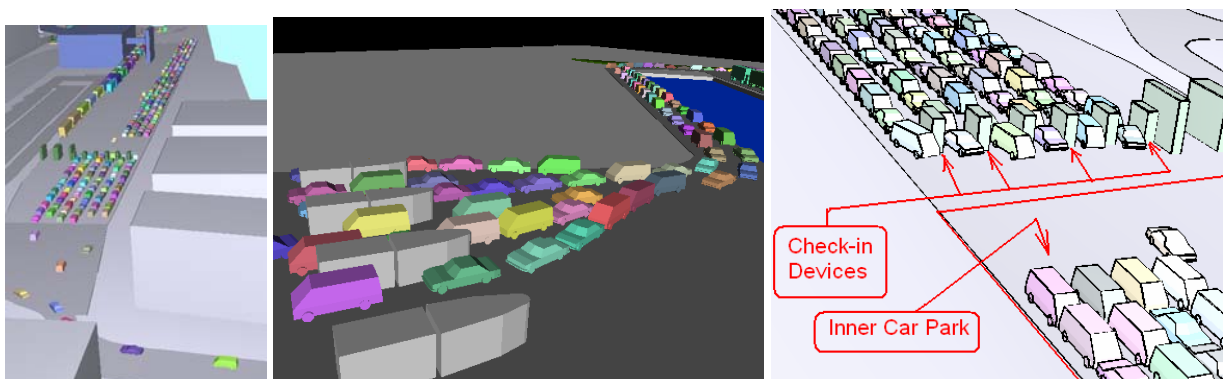


Fig.1: Typical Arrival and Check-in Process (Screenshots from *Helios-PortSim*)

2. **PASSENGERS CHECKING-IN:** As the passengers arrive at the port they take a diversion from one or more public roads to enter into the port area. Then they reach the check-in devices where they use their magnetically or optically coded travel-pass to grant access to the inner car-park where the checked-in cars are parked. This applies to all types of vehicles

except the trailers that arrive much earlier (and are assumed to be already parked in the trailer yard). The time to complete the check-in procedure varies from person to person and is generally assumed to follow a normal distribution. In a busy period during the arrival, there would be queuing of cars at the check-in devices (as shown in Figure 1).

3. **PARKING IN THE INNER CAR PARK:** There is usually a large parking space in which the cars are parked after the drivers complete the check-in procedure. These cars are ready to be loaded once the ferry finishes unloading. There can be several ways that this space can be filled. Normally the operators choose to fill this space one lane at a time so that this parking order can be used to ensure that the earlier one comes to the port the earlier she gets to board the ship and hence the earlier she gets to disembark and leave the port on reaching the destination. This incentive helps maintain a good spread of arrival time.
4. **UNLOADING THE SHIP:** Once the ship is docked, one or more link spans establish a continuous connection between the ship and the port. By that time all the onboard car passengers are in their respective vehicles and waiting for their turn to disembark. When disembarkation begins one or more lanes are signaled to disembark at a time, and this continues in a sequence until the vessel is empty.

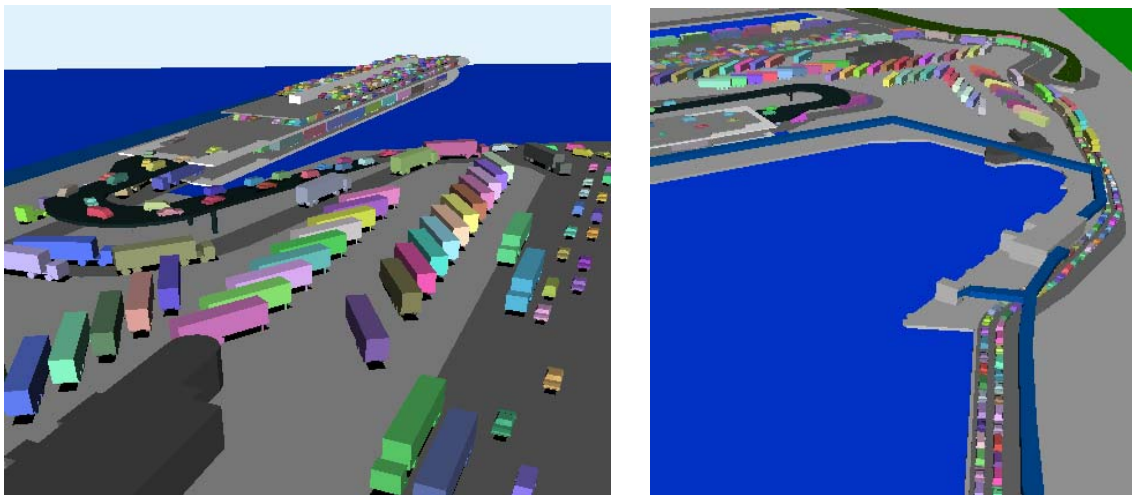


Fig.2: Vehicles disembarking from the ship and leaving the port. (Screenshots from *Helios-PortSim*)

5. **LOADING OF CARS:** The loading of cars ensues as soon as the unloading process is complete and continues until all the cars that checked in have embarked on the ship. Ground and onboard crew coordinate to guide the process.
6. **LOADING AND UNLOADING OF TRAILERS:** Ferries carry a significant amount of cargo on wheeled trailers that don't have their own engine and are driven by the so-called tugmasters. Usually the trailer loading process is the longest sub-process of the loading process, and more is to be gained by optimizing this process. The procedure for loading and unloading of trailers normally consists of the following – The tugmasters unload the first several trailers from the ship (typically twelve or so, for a ship with two trailer lanes on each side of the centre casing). This makes enough free space for loading/unloading to be carried out simultaneously in unloading the rest of the trailers. This is quicker than fully emptying the trailer space and then starting the trailer loading process. This method also has the advantage of affording better utilization of the trailer yard space. After a trailer is loaded (i.e. removed from the trailer yard) its space can be used for parking the next unloaded trailer so that no additional space is necessary for the subsequent process.
Two tugmasters with trailers can operate independently on either side of the center-casing (the center casing is a structural element comprising lifts and stairwells that lies between the starboard and port side of the vehicle deck) but usually there is not enough space left for a

third tugmaster with a trailer to operate simultaneously on the same deck. Therefore sometimes some tugmasters would have to wait for another tugmaster to come off the ship. The tugmasters are very agile and maneuverable when not attached with a trailer. So after detaching from a trailer on the deck, a tugmaster can reverse itself on the same side of the center-casing. The tugmasters make similar reverse maneuvers, in the trailer yard while making their way to other trailers.

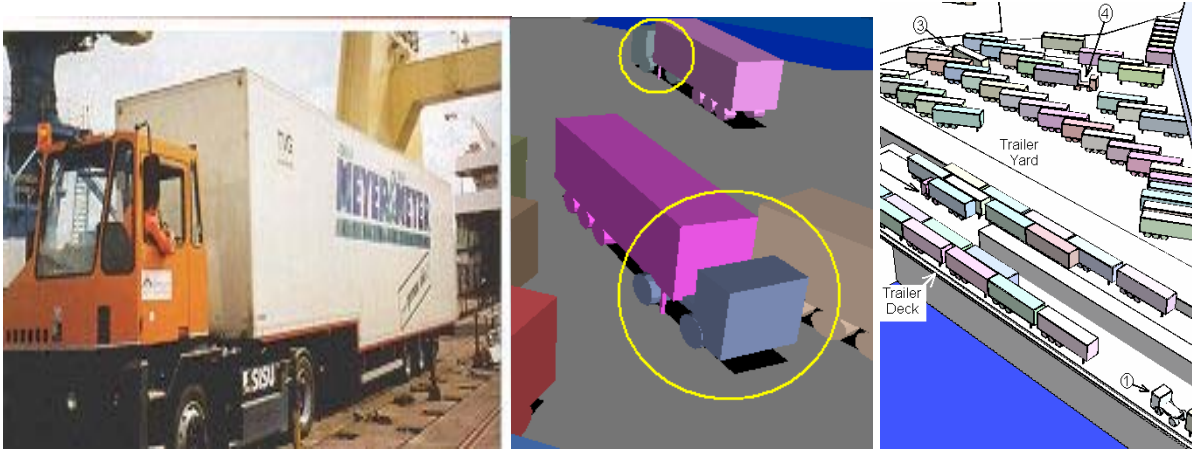


Fig.3: Tugmasters deployed to load and unload trailers.

7. **CUSTOMS INSPECTION:** The customs inspection is a process carried out near the exit of the port. In some ports there is a separate exit lane going through an inspection zone for vehicles that have some items to declare. Besides some vehicles are chosen from the rest of the vehicles for inspection. The customs check of a vehicle within an exit lane causes its tailing vehicles to pause for the duration of the inspection.

The aforementioned procedures are the most significant elements of the loading/unloading process. There are some more sub-processes such as (i) lowering of the mechanical ramps, (ii) attachment of the link span to the ship's bow/stern door, (iii) opening of the bow/stern door, (iv) lashing of the trailers and trucks, and so on. Some of these operations are done simultaneously with some other processes and the rest must be done sequentially.

The primary requirement of *PortSim* was to be able to simulate the aforementioned processes and generate data on the performance of a modeled operation.

3. Design of the PortSim Software

The following subsections briefly describe the key elements of the design of *Helios-PortSim*.

3.1 The PortSim Environment Model

The primary design decision in the development of *PortSim* was that of the level of abstraction at which the operations were to be modeled. The motions were decided to be modeled purely kinematically (i.e. in terms of the geometry of motion, not in terms of the forces causing the motion). *Helios* already had visibility based and graph based motion (planning) model for simulation of pedestrians but the kinematics of wheeled vehicles were fundamentally different. The kinematical models of wheeled vehicles are differential equations with a special property called non-holonomy – referring to differential constraint(s) between kinematical parameters that can not be integrated into an algebraic equation. Although wheeled drives have been ubiquitous for a long time, the kinematical planning of wheeled drive has not been studied until recently [MURRAY, M.; SASTRY, S. (1993)]. The said work and its sequels were studied in the initial phase of *PortSim* development but the high mathematical sophistication of the method was considered beyond the resources available for this project. If we could incorporate the methods presented in [MURRAY, M.; SASTRY, S. (1993)] into

Helios-PortSim, we could have computed kinematically correct maneuvers (motion plans) of wheeled vehicles in the presence of obstacles.

So, instead of trying to compute the motion from scratch, we decided to constrain the motion to a network of pre-defined tracks, and a set of pre-defined maneuvers. The operations could then be modeled using a *task-level* programming of the vehicles. Thus, the instantaneous motion of the vehicles would be 1DOF motion along paths in a 3D network. The *non-holonomic* constraint of the vehicles' motion would then condense into a *minimum-curvature* constraint on the construction of such paths. It would be tedious to construct such paths interactively, so PortSim provides features to construct a full connection network from a skeletal description of the network. Besides, there are track array creation features to minimize the effort involved in constructing the network model.

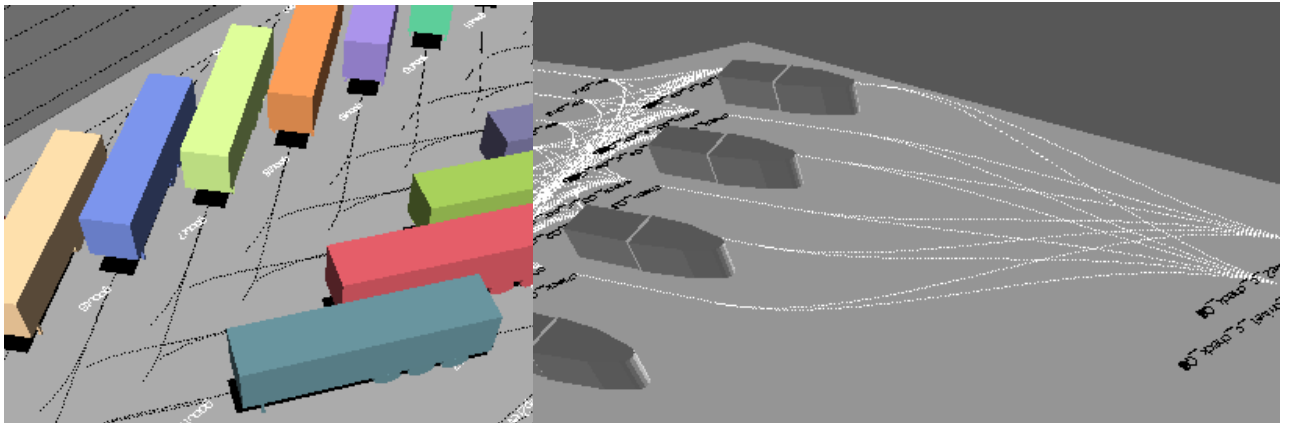


Fig.4: Network of vehicular tracks generated in PortSim, shown as black lines in the left image and as white lines in the right image.

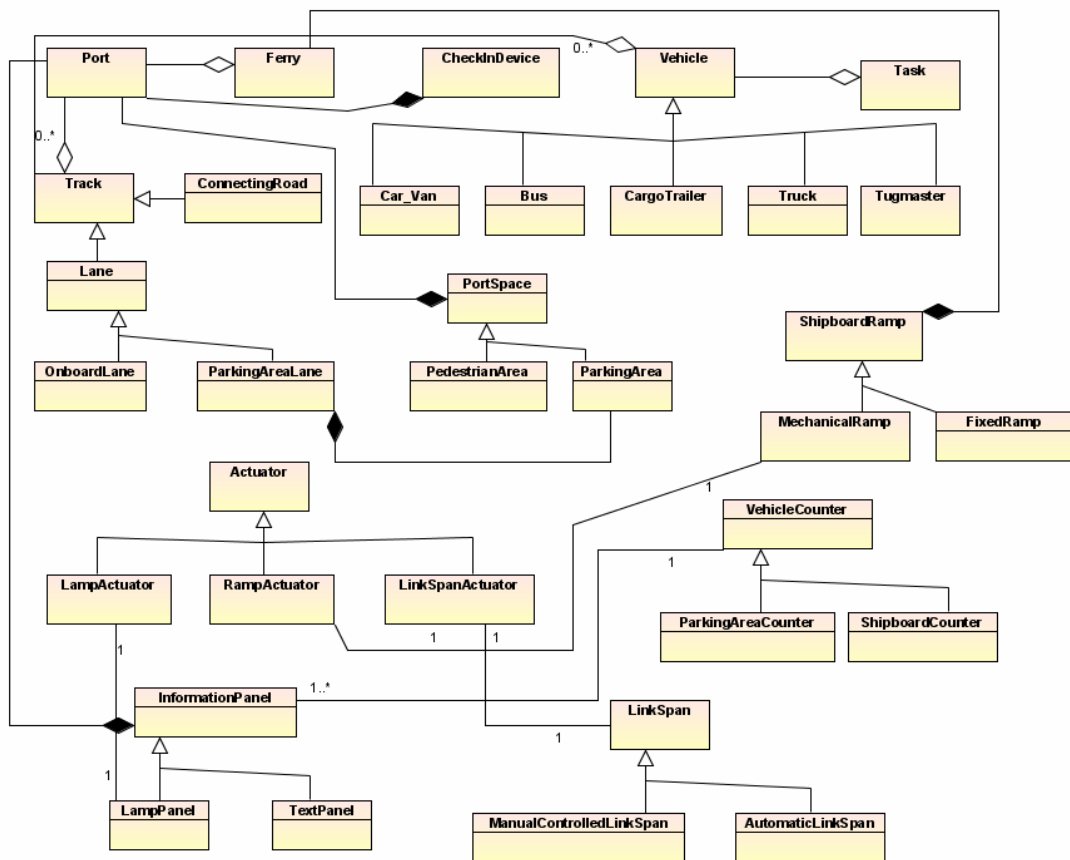


Fig.5: A partial UML class diagram for PortSim.

The aforementioned network model is actually an annotation sketched atop a multi-layer 3D spatial database which forms the core of Helios as reported in [MAJUMDER, J.; VASSALOS D. et al. (2005)]. This database is generated automatically from a 2D map of the port and general arrangement (GA) drawing for the ship.

The path network approach is used in other simulation tools such as Automod and Flexsim, but the path sketching is much more efficient in PortSim due to the facts that (i) node creation is unnecessary, path nodes are inferred from the paths (ii) user only creates a skeletal network and generates the rest of the network using convenient auto-connect and array functionality. The path network so generated may also be tweaked or edited on the popular CAD tools such as Autocad and Intellicad.

Besides the path network, there are a number of other types of annotations that one can do on the Helios spatial database in relation to vehicular simulation. These annotations are sketched on the spatial model of the ship and port environment using simple geometric figures such as circles, polygons, directed lines (arrows), points and rectangles. The sketched figures may be named and used in specifying additional information. For example, one can specify a region using a sketched set of rectangles or polygons and ask PortSim to record the number of vehicles in that region. Such regions are also used in enforcing exclusion zones, sink zones (where vehicles are annihilated once they go beyond the area of interest), and source zones (where vehicles spring into existence during the arrival process). As another example, *one-way tracks* are specified using arrow annotations.

So the environment model for PortSim is essentially a 2-manifold surface in 3D space (as is the spatial database of Helios), annotated with local information associated using overlaid sketches.

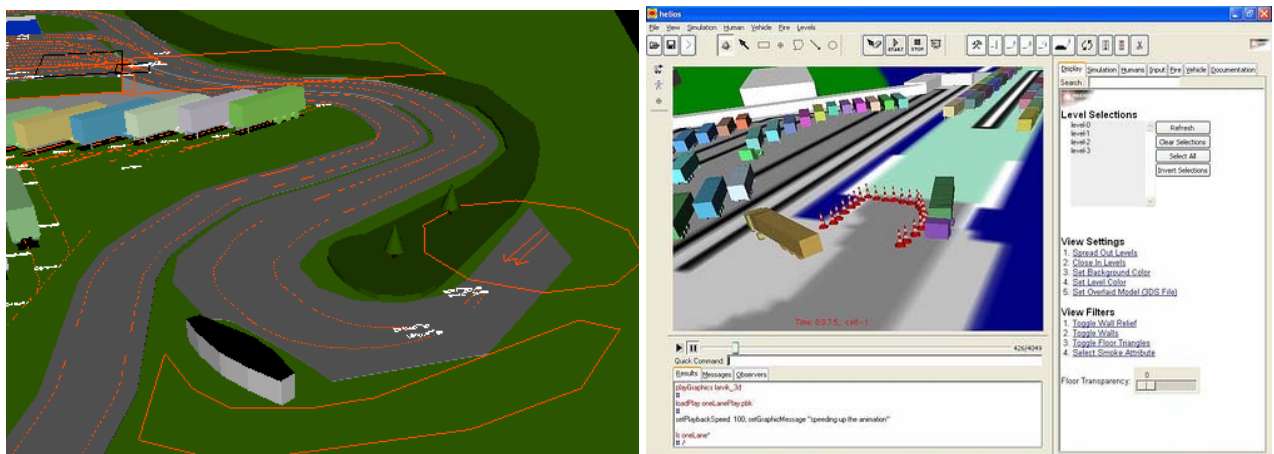


Fig.6: Left: The sketched geometric figures serving as spatial annotations in the environment model. Right: A screenshot of the Helios user interface.

3.2 The PortSim Operations Specification Model

It is usually preferable to make the modelling task a declarative activity – i.e. in this context the model is preferred to be described as a set of declarations about the operational model. This was achieved for cases in which one could declare the final destination lanes for the vehicles; however the specification of operation models has a very imperative semantics, i.e. if the operation is specified unambiguously, such a specification would take the form of a simple task-level program for the vehicles.

This need was addressed by introducing imperative semantics for the description of operations. Thus the tasks assigned to vehicles are not just destinations but a sequence of sub-tasks. Most subtasks are road names but they can be other things such as (1) a maneuver (2) instruction to attach or detach a trailer (3) any script that would be executed with the vehicle's *self* in context (4) pause for a specified duration, etc.

Among the aforementioned sub-task types, *maneuvers* are similar to tracks in the sense that they too are represented by curves and lines in 3D space, but they are interpreted differently. A track is fixed in space and has an absolute position in the port/ship layout. But a maneuver is a path that is followed relative to the current position and orientation of the vehicle. Several parametrically defined maneuvers have been provided in the PortSim data library.

Most of these task descriptions get utilized for tugmasters whose operation happens to be the most complex of all.

The vehicles carry out the list of tasks assigned to them in sequential order, but user interaction and event handlers can potentially change their task assignment. By event handlers we don't mean the GUI event handler for mouse click, keystroke and such. There is a rich set of events available in Helios-PortSim to model a variety of operational aspects.

For example, after driving up the link-span the cars enter through the bow door, where there is a crew member directing the incoming cars into the car-deck lanes, using some policy or logic for distribution. This may be modeled by a handler for an event that gets generated when a vehicle comes to the end of a track that leads up the link-span to the bow-door. This handler is a scripting procedure implementing the distribution policy or logic. Speaking of scripting, all the functionality of Helios is scriptable – i.e. available for runtime scripting using the scripting language called *TCL* (<http://www.tcl.tk>). Thus any event handler can potentially make any change in the state of the simulation. The following are types of events that are detected and handled in the PortSim functionality.

1. A vehicle entering or leaving a road.
2. A vehicle entering or leaving an annotated region.
3. A road becoming occupied, choc-a-bloc full, or empty.
4. A vehicle parking or becoming stagnant.
5. A timer event (scheduled to happen at a certain time or after a certain interval).
6. A periodic timer event (an event that is triggered every so many seconds).
7. Mouse click event at a higher level of abstraction (e.g. with the context of a point in the 3d model space that has been pointed to while clicking, or the object that has been clicked on)

An event handler can do virtually anything that can be done in Helios (e.g. query the current state of the simulation, record information, modify the current state etc.), and it is passed all the specific information relevant to the particular instance of the event (for example, for a road exit event the names of the road in question and the exiting vehicle is provided to the handler). The handling of the aforementioned set of events has so far been sufficient for modelling all the operations mentioned in section 2.

Some features of declarative modelling has been included in PortSim but the aforementioned event handling method has been found to be more flexible in handling all possible idiosyncrasies of the operational specification. Some of the declarative features are as follows:

1. Automatic planning of path to a destination, observing declarative path constraints (one-way etc.) using graph based algorithms. This enables a procedure to be declared only in terms of its initial and final state.
2. Regular expression based specification of actions for cyclic procedures. The planning algorithm for this novel method is very interesting, and is based on deep concepts of computing science. The regular path expression is translated into a finite automaton (which is a labeled graph), and then an automata theoretic product is taken between the path network and the automaton from the path expression. Any path on this product automaton between start and accepting states of the automaton is a path that satisfies the specified regular path expression [MENDELZON, A.O.; WOOD,P.T (1995)]
3. Predefined parameterized event handlers that hide the imperative semantics. For example, a predefined handler “`checkinFillOneLaneAtATime {lane1 lane2 lane3 lane4}`” may be used as an exit handler for a check-in track, that fills the parking lanes *lane1* first, and then *lane2* and so on as the vehicles check in. Such pre-defined procedures are less flexible than an imperative user-written handler but provide a declarative abstraction which reduces modelling effort.

Thus the current design of the operational model may be seen as a partition of the task between (A) built-in features (such as obstacle avoidance and event triggering services), (B) user defined behavior written in the modelling script (C) model data created using sketching features.

3.3 The Data exchange Interface

Helios requires a number of types of input data towards modelling of a simulation scenario. Firstly it requires the layout of the environment expressed as a drawing. Normally a general arrangement (GA) drawing is directly usable if it is available in the AutoCAD-DXF format. If the drawings are too elaborate, they require some simplification. Some additional annotations need to be made in the drawing file (in specially named layers) to specify the bounding polygons of staircases and ramps. Optionally the initial distribution of vehicles (e.g. pre-parked trailers and other vehicles) and tracks may also be done on the drawing. Such annotations are automatically exported into the Helios data files. So, there are two ways of creating Helios annotations - (1) creating within Helios using the sketching interface (2) annotating on a DXF supporting CAD editor such as AutoCAD, IntelliCAD etc.

The basic layout drawing can NOT be built from scratch in Helios, it must come from a drawing file created on a CAD editor.

One can also import colourful and textured 3D geometry files (e.g. given in 3D Studio, DXF etc. formats) for display in Helios. Such models are loaded for display purposes only (e.g. for superposition of the hull, superstructure and the port geometry with the simulation to create a virtual reality (VR) effect).

As regards output, there is no single format that may be stated as the output format for Helios. There are a number of query commands using which the user can get numerical and textual information about the state of the simulation. These queries can be scheduled to be recorded on files either at fixed intervals or in response to certain events or conditions. For example, one might use the query "*numberOfVehiclesInCheckInQueue*" to find the number vehicles waiting to check in. This query may be added to a so called *observer* in Helios, to display its graph plot against time and to record it to a CSV data file which can be opened on a spreadsheet program for post-processing. An example of such recorded data is shown in Figure 7.

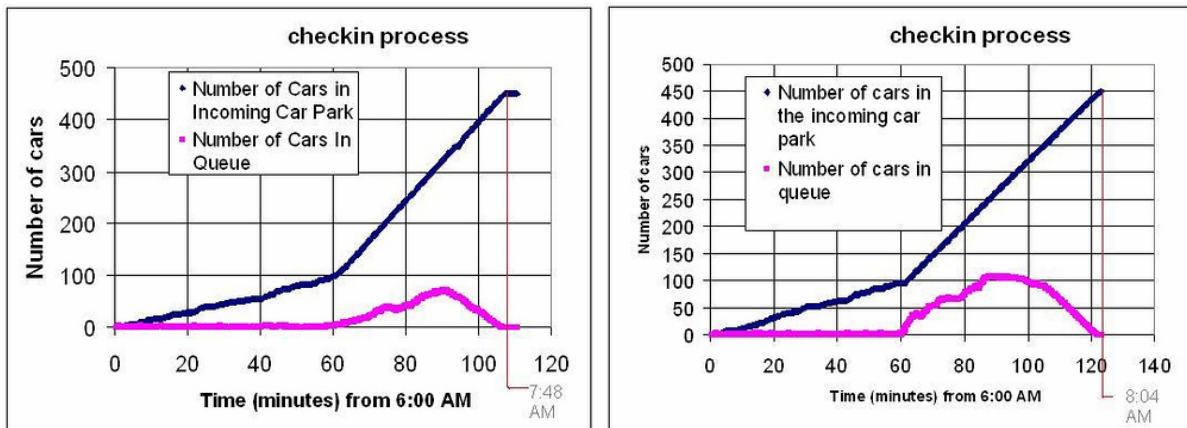


Fig.7: Performance data recorded on PortSim: two operational alternatives for the early morning (8:00AM) departure, the first one meets the schedule by checking in all passengers 12 minutes before the departure, in the second one it fails to check everyone in.

It is up to the user to select the information to be recorded. One can also save a playback file to enable *replay* of a completed simulation without having to re-compute the simulation. It also supports export of the OpenGL frame-buffer into bitmap files for every screen update. This feature came handy in recording the screen animation on a very old computer without 3d

graphics accelerator. The exported images for each frame were later collated to form a smooth animation.

Besides exporting the frame-buffer image, vector graphics diagrams (in the FIG format) may also be saved periodically or in response to events. In the vector graphics export mode, a 2D drawing file is exported with colour coded occupants and vehicles marked within the drawing.

To summarize the data exchange interface of Helios-PortSim, the input data can be created using Helios, a text editor (for scripting), and a CAD editor; and there is a wide variety of output options to choose from.

4 Conclusion and Scope for Future Work

The *Helios-PortSim* tool as described in sections 2 and 3 has been useful in making decisions about the design of port facilities and processes with respect to vehicular operations. It has been efficacious but there still is significant room for improvement.

There are a number of ways in which the Helios-PortSim model can be improved. Firstly it should be possible to make the modelling task largely declarative by providing a parameterized object library for all procedural elements. This should enable the user to pick and place such objects without having to resort to any scripting.

The simulation functionality may also be extended to include additional dynamic models such as (1) multiple ships in the port waters, and (2) container operations involving cranes.

The kinematical model of vehicles can be improved further by introducing the non-holonomic motion planning as described in [MURRAY, M.; SASTRY, S. (1993)].

Currently the simulation data for a given simulation project resides in multiple files. It would be more convenient to the user if there is a single database file that contains all the data pertaining to a project.

It would also be useful to improve the 3d content base for the simulation. For example, the current generic vehicles may be replaced by fully textured realistic meshes. We could provide a 3d object gallery to drag and drop to help the users easily make the simulation visuals more convincing.

References

MAJUMDER, J.; VASSALOS D. et al. (2005) *Modelling and Simulation of Shipboard Environment and Operations*, Ship Technology Research Journal, Volume 52, Issue 4, October 2005, page:159-171

MURRAY, M.; SASTRY, S. (1993), *Non-holonomic motion planning: Steering using sinusoids*, IEEE Transactions on Automatic Control, 38(5) 1993, pp. 700-716

MENDELZON, A.O.; WOOD,P.T (1995)., *Finding regular simple paths in graph databases*, SIAM Journal on Computing, 24(6) December 1995, pp. 1325-1358.