

Passamonte

25th May 2019

1	Intro	2
2	First look into Passamonte	3
2.1	Analysis	3
2.2	Design	4
2.3	Implementation	6
2.4	Marketing, business, viability	6
3	Let's get our hands slightly dirty	7
3.1	How it's gonna be	7

List of Figures

1	Passamonte deployment	3
---	---------------------------------	---

List of Tables

1 Intro

Passamonte (related to Maese Pedro's puppet show, or "*el retablo de Maese Pedro*"; Maese Pedro was also known as *Ginesillo de Parapilla*, or *Gines de Passamonte*¹) is a synthetic environment software (or hardware: see deployment below) solution, to fit a sim (typically a real-time sim for training) with automatic, AI-controlled, rule-controlled, or instructor-directed entities to populate its simulated world. The sim manufacturer gets a synthetic environment with selectable characteristics, and with a big initial library of entity models.

The selectable characteristics are features that can be brought in by Passamonte or the simulator, at the user's choice. This includes

- gravitational models;
- magnetic models;
- atmosphere models (including thermodynamics, winds, turbulence, wide-area weather, CFD for wind around obstacles, et cetera);
- celestial bodies (e.g. for orbital simulation);
- fire models (for fire fighting simulation);
- collision model².

Selectable characteristics are used by Passamonte's entities, but can (and should, for coherence) be queried by the customer's sim.

The library of entity models grows to accommodate more and more customer's needs; it includes

- vehicles (space-, land-, water-, and air-based; free or following roads);
- inert bodies, such as slingable containers and rocks;

¹Cervantes' original spelling.

²Although the general collision model should probably come from the customer, being based on a scene graph, and should be queried by Passamonte through the network, a part of it, namely the one involving models run by Passamonte, should probably remain within Passamonte.

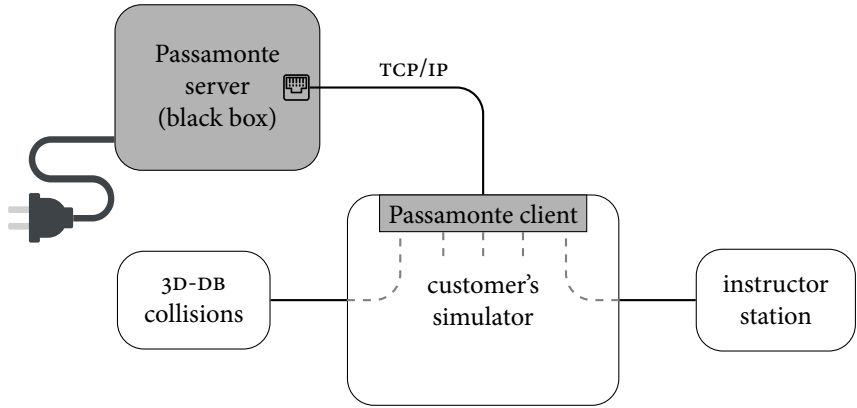


Figure 1: Passamonte deployment

- vegetation responding to wind, rain, or fire;
- humans and animals.

Passamonte is deployed as a black-box connected by TCP/IP (see Figure 1). We deliver the box, branded “Passamonte”, as a hardware component to hook up to the rest of the simulation with a network cable. We also deliver the Passamonte client software, with the option to customise it to the customer’s requirements, software needs, and programming language. The Passamonte client software interfaces with the customer’s software through a regular API. Any communication between the Passamonte box and any other component of the sim is handled by this single client instance, through the computer it runs in³; it is therefore the customer’s responsibility to route communications through the computer he chooses to install the client in.

2 First look into Passamonte

Just enough detail to get the next chapter (§3) going.

2.1 Analysis

Requirements hodgepodge:

- real-time;

³We may have to relax this, if we need fast communication with other components. We would then install a Passamonte client in every such component.

- driven by an external clock (also good for as-fast-as-you-can automatic tests);
- text-based parameterisation (files sent over the network at initialisation);
- extendable with customer's models (code sent in source or binary form over the net at initialisation);
- socket-based interface;
- the socket-based interface must accommodate delays;
 - possibility to subscribe to observables, in both directions, to save one trip (i.e., if i always need the current position of a given entity from another machine, avoid asking for it at every simulation step: just subscribe to it and get the info without having to ask for it);
 - possibility to query about the future when feasible (turbulence, free waves, motion of big vehicles [e.g. aircraft carriers] not affected by small ones [aircraft], slow-dynamics processes, et cetera);
- high-quality physical models;
- high-quality contact models:
 - Runge-Kutta integration;
 - up-to-date interactions between Passamonte models;
 - ray-cast-based visual database queries;
- selectable model accuracy for in- and out-of-focus situations;
- selectable model accuracy for special situations:
 - many entities;
 - low-detail data available for parameterisation (e.g., kinematic models may be OK if we only have flight manual charts for the parameterisation);
- independence of order of execution (compute-state separate from update-state);
- questions and answers across the net always tagged with time;
- simple visual system with “god’s eye” (ability to move a camera around);
- monitoring system.

2.2 Design

Outline (in bold, required for the MVP; see §2.3):

- framework

- **sim core**
 - * **model tree, model execution**
 - * **messages across the tree**
 - * **text-based communication with the server (creation, destruction, query, observation, commands ...)**
 - * **state, observables**
 - * **numerical integration: Runge-Kutta big-vector integration**
 - * **generic text-based I/O for values;**
 - * **interactive and offline monitoring**
- **basic libraries**
 - * **parameterisation**
 - * **magnitudes with dimensions**
 - * **filters**
 - * **geometry**
 - * **intersections**
 - * **focused simulation**
 - * **dynamic loading**
 - * **model testing and validation**
 - * **visual system**
- **domain libraries**
 - * **Newtonian mechanics**
 - * **flat-earth gravity**
 - * **aerodynamics**
 - * **contact**
 - * **thermodynamics, engines**
 - * **systems (electrical, hydraulic ...)**
 - * **detection**
 - * **autopilot, synthetic pilot, guidance**
 - * **AI, entity coordination, strategy**
- **communication, for server and client**
 - * **stream based questions and answers**
 - * **file passing**
- **dynamically linked:**
 - **selectable characteristics (this should grow as needed; see some initial examples in §1)**
 - **entity models (ditto)**

2.3 Implementation

Just some pointers for implementation.

Implement the net communication first, since it's the highest risk; test for flexibility, delays, subscriptions, creations, destructions, et cetera. Investigate real-life sources of net problems. Every development must be autotested over the net, even if it is in a single computer, with simulated net problems.

Use a scene graph from the beginning, to see what we're doing, and to have terrain and intersections, which are gonna be very important. But, as always, everything must be autotested.

We'll need to have most of the library dynamically loaded, since it's important to give the customer the opportunity to load his own libraries at run-time.

The customer installs *nothing* in our box. That way, we always know exactly what's in it. This is why we need to be able to pass binary libraries over the net, that get dynamically linked at run-time.

Check the idea of focused simulation. It should work, look good in the visual system, and give consistent results across the focus.

Minimum viable product (MVP): bare-bones implementations of bold items in the outline in §2.2, plus the following (also bare-bones):

- selectable characteristics:
 - flat earth;
 - constant gravitation;
 - simple non-flat terrain;
 - simplest viable collision model;
- entity models:
 - big non-articulated truck with a heliport of sorts on its back;
 - simple drone to land on the back of the truck (no aerodynamics; just thrust).

2.4 Marketing, business, viability

Marketing emphasis on closed, black-box solution, pluggable into the customer's network. Very easy to deploy for the customer (adapt the client, et voilà).

Clients:

- obvious clients: KMW, CAE, Thales, Indra, Airbus...
- not so obvious clients: smaller simulation companies, who may need ready-made solutions ThatJustLookGood™ to populate their simulated worlds.

The viability would be high if we can pull a EU R&D grant. We must research this possibility⁴.

I think we could get a MVP in six months (what a blatant lack of precision; see §3.1 for a little more commitment).

3 Let's get our hands slightly dirty

3.1 How it's gonna be

The changelog for Passamonte, after successful completion of a MVP, will look like this:

```
* | [mkt1] check grants
* | [mkt2] start market R&D and contacts
| * [cor1] basic core backbone, just enough to start [lib1]
| /|
| * | [lib1] initial offline library: dynamic loading
| / /|
| | * | [vis1] 1st visible things: flat scenario, truck, drone, tree
| | | * [cor2] model execution, trees, state, comm thru the tree
| | | * [cor3] server-client comm framework
| | | * [cor4] textual representation, interactive monitoring
| * | | [lib2] tables, geometry, abstract interface for intersections
| | \ |
| | * | [vis2] concrete interface for intersections
| | * | [vis3] more graphical detail, terrains
| \ \ |
| * | [lib3] magnitudes, filters
| | * [cor5] full state vector integration
| * | [lib4] filters
| \ |
| * [cor6] completion of comm framework; a lot of testing
| /|
| * | [com1] stream comm and subscriptions
| * | [com2] files through stream comm
| / /|
| | * | [dom1] Newtonian mechanics & gravity
| | * | [dom2] very simple action generators: torque, lift, drag
| | * | [dom3] contact forces
| | * | [dom4] truck & drone with autopilots
| \_ \ |
```

⁴Take a look at <https://arosaid.com/2019/05/16/cdti-programa-neotec-2019>; there are companies who help you with the bureaucracy, e.g. <https://www.zabala.es>.

```

|      * [lib5] focused simulation
|      * [mvp1] first MVP, packaged into a black box
|_____/|
*      | [mkt3] 1st tournée showcasing the technology
----- 6 months
|      * [cor, lib, vis, dom, com] the development goes on...
*      | [mkt] the market R&D and the contacts go on...

```

So, this is what i think can be achieved by (not any) four persons in 6 months (see §2.4).

4 Backlog; ignore, probably

Notes about the development of a synthetic environment for simulation, packaged as a marketable product. Alternative names: titilimundi, retablo, Maese Pedro, Ginesillo.

Ideas to research, things to describe, tasks:

- R&D about the competition; i don't think Havok should be seen as competition⁵ (or Unity3D, or Ogre3D); maybe we should look at *massively multiplayer* environments;
- focused simulation: only models that can be perceived by the user are simulated with full accuracy; ramping in and out of focus;
- querying about the future (but not too much); network delays can be countered in some situations by querying not about the present state, but about the state after a small number of frames; this has to be tracked, formalised, and investigated;
- the sim's models are mirrored by *ghost entities* in Passamonte;
- should we add the possibility to host even the sim's models in Passamonte? they could be sent in source or binary form over the net; this may simplify complexities added by the need for distributed model running;
- several alternatives for the deployment:
 - software
 - * with a regular C++ API, and possibly bindings to other languages,
 - * or with an HLA (or DIS) interface;
 - standalone computer connected

⁵Havok is not even meant for simulation. There must be other, more à propos software solutions for synthetic environments for simulation.

- * through an HLA (or DIS) interface,
 - * or through a specific network interface, with a small client software installed in the customer's computers (this alternative seems the most promising to me);
- where to cut:
 - terrain; how do existing scene graphs offer terrain information?
 - * OpenSceneGraph: an `Intersector` intersects the 3D database in a specific way, and returns information about the intersection; there are several kinds of intersector, depending on the object that intersects the database: line segment, plane, polytope; for a intersector representing a line segment, the returned intersection information, with type `osgUtil::LineSegmentIntersector` contains, among other things, the intersection point and the normal at the intersection point; other elements seem to point to the intersected object, so more information could be obtained (like, e.g., material);
 - * Unity3D: a ray cast (`Physics.Raycast`) returns information about the hit (`Physics.RaycastHit`), containing, among other things, the intersection point and the normal at the intersection point, and a pointer to the intersected object;
 - * Ogre3D: i didn't investigate its implementation;
 - atmosphere: should probably be included: i have the knowledge, and even good novel ideas; the atmosphere can arguably be considered a part of the synthetic environment; same as sea waves, BTW ...; celestial orbits too; plus, the atmosphere model can be extended to become "Susuwatari";
- parallel computing:
 - CUDA,
 - OpenCL,
 - maybe check Nvidia PhysX and similar frameworks;
- check available socket communication libraries; there may be ready solutions for what we need (stream-based communication, passing whole text or binary files at the beginning, passing small parameter files any time ...).

Important non-technical tasks:

- check EU R&D grants; there may be something to finance Passamonte;
- check also the Spanish military COINCIDENTE programme.