Hovercraft Telemanipulation Virtual Models and Interface Design

PROJECTO DE MESTRADO

Hildegardo José Quintal Noronha
MESTRADO EM ENGENHARIA INFORMÁTICA

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

Novembro | 2010

Abstract

A control system was designed to allow humans to manually drive an, usually automatic, two wheeled hovercraft. The size, the mass and the way of driving this vehicle proves to be an issue for the everyday, untrained person to achieve.

During this thesis several control layouts were designed with the objective of creating an intuitive and easy way of driving such a vehicle. At the end two where user-tested using a simulation (also developed during this thesis) of the said hovercraft set against obstacles similar to those expected to be encountered on its real environment.

The two layouts are just slightly apart in performance but numerous issues were found that can be used to redesign a better control layout. This means that no definitive winner was found but a foundation for a better design was indeed found.

Contents

Introduction

Context

The *ITER Project* is a big international research and development project to build a fusion reactor. The reactor and its support buildings are being built in France, inside which a vehicle will move material around. This vehicle will move through the dangerous and radioactive reactor room where no man can enter.

This vehicle is massive (as big as a bus) with the ability to move as much as 100 tons. It will move through tight corridors and make sharp turns. Such a vehicle must be agile leading to a strange design. The current design points to a vehicle similar to a hovercraft. It has two wheels, one on the front and another on the back. Those wheels have the ability to turn 360 degrees and have traction capabilities. This makes up to a very strange vehicle by every day's standards. The good part is that this hovercraft is not supposed to be driven by humans. But a vehicle running around such an important and dangerous location where humans cannot enter should not be without a backup system to fall back to. It is that system that this thesis tries to build: a way of controlling the strange vehicle using humans instead of a computer program.

Goals

A software system is to be built to serve as a test environment for further research and development. This software is a physical simulation where the user can drive a 3D mock-up of the real hovercraft through an environment. It is also the responsibility of the simulation to log several parameters of the simulation so this data can be used to study how users handle the hovercraft and different control layouts. A second program needs to be developed to handle the hardware (explained on the next paragraph) and send its information to the simulation. Yet another program has to be coded to extract information from the logs created with the simulation, during the user tests, and make it human-readable.

A hardware controller needs to be developed and physically prototyped to be then used and tested with the simulation. This controller, despite not being very high-fidelity, needs to be in working order and to handle as expected so users can behave as close as they would with a final version.

A final test needs to be deployed with several users so data can be extracted on each control layout and then analyzed to find the best control layout (of those designed) as well as issues and possible improvements. This is also the best point to study how users react to the concept of the hovercraft and

how they expect it to be controlled and to receive verbal (and other kind, such as emotional) feedback.

Contribution

The simulation built can be used to test any kind of new hardware and any hardware combinations. It is very easy to change the code to cope with the different APIs that handle different hardware. It can also be used to train users on a controller layout and far from danger of physical damage or, if the simulation accuracy is not enough as desired, it can, at least, give a first contact with the system. The simulation can, however, be perfected and more accuracy can be added as well as more components to fit the needs of the user.

Two control layouts were built and tested using 11 users. These tests create a base from which future work can be developed since it delivered a first contact with this vehicle way of handling, many issues and pointers were raised. Those issues can be avoided in future work and the pointers can aid developing a better control layout.

The user test did not only give feedback on the controls. It gave feedback on how the users are able to grasp the hovercraft's concept and how easy they learn how to control it. This can be used to aid the refinement of the hovercraft design itself which can have consequences on the way to control it and, ultimately, on the controls layout. User's expectations, needs and way of handling the simulation's hovercraft can also be incorporated on a more pleasant and ergonomic controller.

Summary

The first chapter – *State-of-the-art* – deals with the current state of the art. It reviews the available technology to create the context and aid on the decisions to be made on this thesis.

The second chapter – *The ITER project* – introduces the international project. It gives general knowledge of the project and more detailed knowledge on the parts relevant for this thesis (parts like the reactor and the vehicle to be used inside it).

The third chapter – *System Architecture* – contains the decisions made to build the software system that supports both the simulation and the hardware interface.

The fourth chapter – *The Simulation* – deals with how the simulation and all the related artifacts were built.

The fifth and last chapter – *The Controller* – details the physical prototype of the controls, how it was designed and the results of the user tests.

The appendices contain data needed on this document. Not all the data produced are presented on this thesis. The missing data is on the CD that accompanies this thesis as well as better versions of the same data (higher resolutions images, excel files instead of extracts and more).

1 – State-of-the-art

This chapter will focus on reviewing the current relevant technology available to date. It helps to contextualize and to make decisions on the project.

1.1 – Control Theory

Controlled systems exist on the nature and as man-made. On the nature, and as an example, living beings need to regulate their temperatures. A man-made example could be a home refrigerator regulating the temperature or a cruise control on a car regulating its speed. The focus here is the man-made systems.

To control a system, first the system needs to measure its own output and then decide how to react. This is called *Feedback Control*: "Feedback control is the basic mechanism by which systems, whether mechanical, electrical, or biological, maintain their equilibrium or homeostasis." (1)

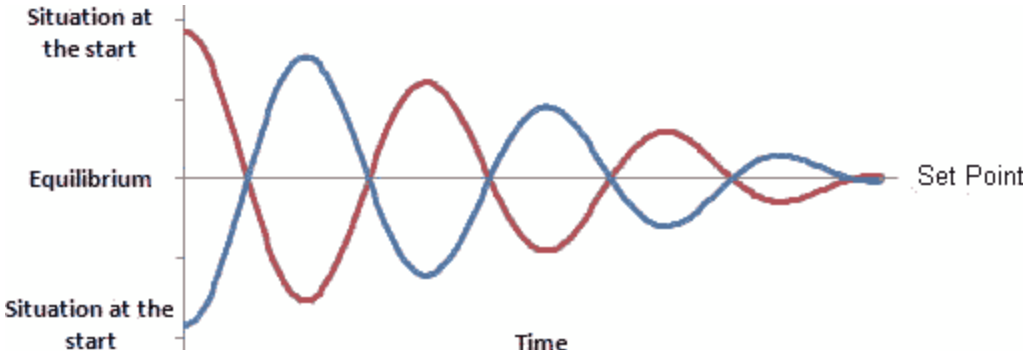
The image to the right is an example of a controller. It uses the gravity and centrifugal forces to control a valve that accelerates or decelerates a steam engine. The faster the engine moves, the faster the controller rotates and, consequently, the higher the balls rise. This closes the valve and decelerates the steam engine. If the engine slows down, the balls are pulled down by gravity, opening the valve and accelerating the engine.

Img. 1 - Watt Governor; A control mechanism used on steam-engines (79)

The component of the system that is responsible to make it behave stable is called a *Controller*. Designing and analyzing existent systems is the responsibility of a Control Theory engineer (2). More specifically, the engineer deals with the controller. The better the controller is designed, the better the system will perform. But if the engineer does not build and tune the controller carefully it can cause the system to behave unstable, erratically or even dangerously.

1.1.1 – Negative Feedback

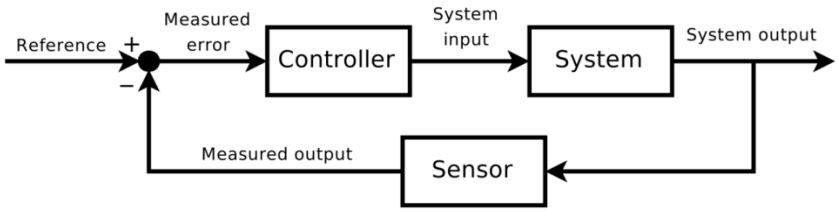
One way of achieving the previously stated system stability is by means of Negative Feedback (2). Negative feedback is no more than reading the system's output and trying to compensate it in a way that the system will, in the future, get to a stable state or state of equilibrium - and stay there. The name comes from the fact that if the system is moving in one direction, the controller will try to make it go on the opposite direction, trying, this way, to achieve the said state of equilibrium. This state of equilibrium is an ideal state which can never be achieved completely. So, the control has to constantly work to adjust the system so it stays closer to the equilibrium state.



Img. 2 - Negative-Feedback illustration. The system output is countered by an opposed input, trying to get the system to the equilibrium point. (2)

As an easy example of a system using negative feedback, the home refrigerator tries to stay at the same temperature set by the user. This works by measuring the current temperature and counteracting it. If the temperature is too high (hot) the thermostat turns on the fridge's cooling system making it cooler. If the temperature is too low (cold) the thermostat shuts down the cooling system so the temperature may rise again. This temperature will oscillate around a set point, always trying to get to that set point which is called equilibrium point.

The implementation of Negative Feedback, on a system, is achieved through the following model (2):



Img. 3 - Feedback loop (2)

There is a system that makes an output. That system is given and is not to be modified. Using the previous example, this system is the cooling system. It outputs cold or a negative relative temperature. The system's output is measured by a sensor (also not to be modified) and then feed back to the controller (this is where the engineer should work). This sensor can be the fridge's thermostat. But the value sensed is not, directly, feed to the controller. It is first subtracted from the value that the system is receiving as an output. For instance, our home refrigerator will start "producing cold" if the temperature is too high and will stop producing it when it gets too low, oscillating around its equilibrium state.

1.1.2 – PIDs

A PID (Proportional Integral Derivative) (3) is a control algorithm that controls one input based on one output of the same system. As stated by F.L. Lewis: "N. Minorsky [1922] introduced his three-term controller for the steering of ships, thereby becoming the first to use the *proportional-integral-derivative (PID)* controller". (4)

Each component solves a part of the problem. The *Proportional* (5) component sets how fast the system will get the set point. It reacts proportionally to an input (if the set-point is close it will react weaker than if it is set farther from the current state) and, as such, is a rudimentary way to control. The *Derivative* (5) component is used to accelerate the reaction of the controller by overshooting. Too much overshoot can increase the time the system takes to stabilize, making it worse than if not using this component. Finally, the *Integral* (5) component is used to reduce a steady error.

A PID needs to be tuned to every system and it is not easy to do it. There are several ways (6) to tune a PID but they usually require some work and/or time. You can tune the PID by hand, basically using trial an error. This usually takes too long and gets lousy results. There are algorithms that can find a starting point from where to fine-tune, if needed. And there are software/hardware (7) that can automatically tune the PID. Also, some PIDs keep auto-tuning during its life cycle.

1.2 – Degrees of automatism

A system can be operated by humans, by computers or a by mix of both. The levels of computer assistance can range from fully manual – the human operates the system on his own – to fully automatic – the computer operates the system on its own (8). The levels in between are achieved with a compromise between human and computer operation.

This subchapter is based on the chapter *Automations* from *Human Factor Engineering* (8) and its definitions of the different levels of automations.

1.2.1 – Completely Manual

This kind of operation requires the human to decide and act upon the system. The human is not to expect any kind of aid from a computer. This level gives the most control to the human but can be tedious or very hard to complete which can lead to incidents.

1.2.2 – Completely Automatic

Automatism is usually used because the task to be performed is either hazardous or simply impossible to humans (for instance: to drive a vehicle on Mars). It can also be used to ease a difficult or tedious task, such as on an assembly line. Some automatisms can simply be used to extend human capability and not replace it altogether. Finally, the technology can be so cheap or mature that automatism simply gets used.

On this operation level, the computer takes full control of the system. The human is completely ignored and all the decisions (correct or not) will be made and carried by the computer. On this level of automatism the human is free to do other tasks but, as with the other extreme (completely manual), this kind of operation has its problems. Usually an automatic system carries its responsibilities quite reliably. But when problems occur it gets harder to fix them in time to avoid incidents (the human has no way to act upon the system).

1.2.3 – Cooperative

The previous two points stated extremes on the way of controlling a system, giving total control to either humans or computers. Also as stated both have their issues. But with a level of cooperation between humans and computers, the system can be operated more efficiently. Depending on the system and tasks to be completed, more or less automatism should be used.

Basically humans and computers can decide and act but only one at the time. If the human is to decide and act, the computer can suggest decisions and point its benefits and consequences. This is called Decision Support. If the computer is the one deciding and acting, the human can either be just monitoring (called Monitored Artificial Intelligence) at which level the human can veto the decision but cannot change it directly. Or the human can be given some choices and, if ignored, the computer will decide on its own (Consensual AI).

Between those levels, we can find any granularity needed to the system, giving more or less power to either human or computer to find the correct balance for that system.

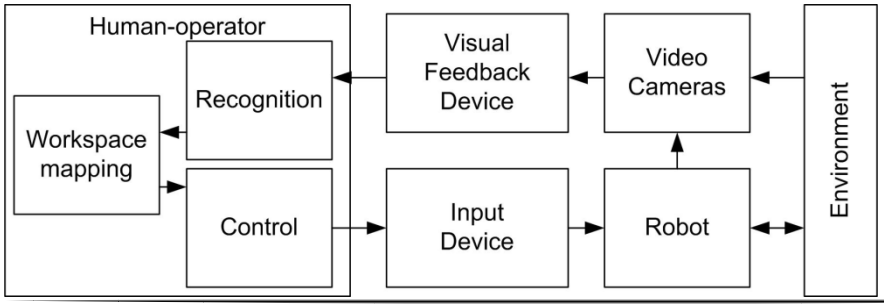
When a system has less help from a computer than it should, the human will fatigue faster leading to less performance and, possibly to incidents. On the other side, if too much automatism is used, the human can find the task too easy and lose his arousal, leading to distraction. This makes the human take longer to respond to a problem which would not happen if the human was on full control. The human can also develop a sense of trust on the computer overseeing small issues which the computer may be compensating to until it gets out of hand and a bigger incident happens.

1.3 – Awareness

Awareness is a synonym of alert, watchful, vigilant (9). It also means that you "have knowledge of". More specifically, Situation Awareness "is the mental representation and understanding of objects, events, people, system states, interactions, environmental conditions, and other situation-specific factors affecting human performance in complex and dynamic tasks" (9). This is critical on teleoperation where the success of the operation depends and is improved by the quality of the situation awareness (10) (11). The user needs to know the state of the robot and its environment before making good decisions. If the user does not know, he cannot decide. If he cannot decide, he cannot perform. Take for example a Search and Rescue robot that has a camera with a very narrow field-of-view. The operator will miss much of what is around the robot and may miss survivors. He may even get the robot stuck in one obstacle just because he was not aware of it (during the tests in *Improved Interfaces for Human-Robot Interaction in Urban Search and Rescue*, they noticed that the robot was crashing into obstacles because it had no camera on the back (12)).

1.3.1 – The importance of Information

The development of computers and the rise of the Internet gave birth to the Information age. This is the age where information is the most valuable asset. It can build an empire or take it down. In robotics and, more specifically, in teleoperation it is not any different. Information is an important part of any teleoperation session. Information needs to be transferred back and forth for a teleoperation to succeed. If the information does not get to the human or from the human to the robot, control will be lost over it. The same can happen with corrupted or out-of-time/out-of-order information. The last two can even be worst in a sense that there is information but the information is bad. If the problem is not detected, bad decisions by the human or bad actions by the robot may be taken.



Img. 4 - The cycle of information on a teleoperation session

On a normal teleoperation session, the information is gathered from the environment using a number of sensors, it then travels back to the human through some channel and it will finally be presented to the user in some way. This allows the user to make a decision and send the input to the robot which will act on its environment through the use of actuators, closing the loop.

1.3.2 – Environment

Environment can be defined as "external conditions or surroundings" (9). Extending the definition, it is everything surrounding the object or living being in question which it may interact with. In teleoperation, the environment pays a great role. The robot is moving around an environment and acts upon it and artifacts located on it. Even robots that do not move around (remote surgery, for instance) or robots that move but do not act (unmanned aerial vehicle as an example) need to be aware of their environment because even when not acting upon the environment, it may act upon the robot (the roof falling on top of a search and rescue robot). On the human side, if the human does not know the environment, he cannot move around and make good decision. Decision made not taking the environment into account may end up on a failure of the task/mission or a complete loss of the robot.

To be aware of the environment, the robot needs to sense it. The following section discusses sensors, which are used to that task.

1.3.3 – Sensors

Sensors are devices that measure a stimulus (9). They are used in about anything that requires information to be gathered from the environment (thermometer, cameras and accelerometers are just a few examples). They are used on teleoperation to inform the robot and the human operator of the configuration of the environment (for instance, a camera can be used to show the human what is on the front of the robot).

Following is a (non-exhaustive) list of possible sensors to be used in teleoperation, separated by function.

1.3.3.1 - Location

Geographic Location – The geographic location can be found, more or less accurately by using direct or indirect measurements. Directly by using Global Positioning System (GPS) (13), indirectly by using, for instance, Global System for Mobile Communications (GSM) towers (the towers used on mobile phones communications). A mix of GPS with assistance from GSM towers is called AGPS (Assisted-GPS). It is getting easier and cheaper to locate and improvements or even new ways get invented regularly (13) (14). It can be used to very serious reasons, such as knowing from where an emergency call is coming to pure ludic reasons, like tagging a photograph to the current location.

GPS is gaining usage as more and more devices become location-aware (14). It is a small chip that receives signals from geo-stationary satellites and, based on the position of at least three satellites it can calculate (triangulate) the chip's position. The more satellites it uses the more accurate the position is (13).

A cheaper solution, specially for mobile phones, is using the GSM towers. Each tower has its own geographic position so a mobile phone can calculate an approximate position based on what towers it can detect and the distance to them (calculated using the time delay of the communications) (15). This solution isn't any way near, in terms of accuracy, to a GPS. But it is more than enough when the device only needs to know in which general area it is. Plus, it can use already deployed resources - no additional cost.

A GPS should be used to accurately resolve a geographic location (it has an accuracy that averages the 15 meter – better resolutions can be achieved but are locked for military usage (13) (16)). GSM triangulation should be used as a cheaper solution if the user only needs a general area location (if used alone it can get a resolution from around 500 meters to many Km – tower's range).

Area Probing – There is a kind of sensors that allows you to know the distance between the sensor and one or more obstacles. Some even allows you to know some more characteristics about the obstacle.

A laser can be used to, very accurately, measure distances (17). It uses electromagnetic waves as its carrier (it works close to the speed of light) so it can measure things very far and very rapidly. Accuracy can also be the down point of lasers: since it can only measure a point it isn't any good if you need to know if something is near or in an area/volume. It can, however, measure multiple points if the laser is aimed at different positions.

Sonars use sound wave to detect not only distances/positions but also speeds (18). Positions are calculated by sending and waiting for a return sound (echo). The speed of an object can be measured using the difference of its past positions or the Doppler Effect. Sonars are less accurate and slower than lasers but can measure in a wider range as well as multiple objects at the same time. It is also a cheaper solution for the following technology, the Radar.

Radars, like the sonars, can also be used to locate objects in range (19). Though, the technology used is different. The radar uses electromagnetic waves to achieve this instead of the sound waves used by sonars. Radars can as well detect objects in a wide angle and use the Doppler Effect to detect object's speeds. Radars are much more accurate and have a much longer range than sonars but don't work well through dense materials (because the waves are reflected). With further processing of the signal, the radar can extract more information about the targets. It can detect, not only the position and speed as well as size, direction of movement, and even material (for instance it can separate rain from a plane).

Lasers should be used to accurately measure distances of unobstructed distant points. But if the point is to detect obstacles nearby, sonars should be used. If there's a need to detect and monitor multiple objects in a range around a point, sonars or radars can be used. There's a preference to radars if above water or in space and sonars if under water.

1.3.3.2 – Local Orientation

There are sensors able to give the local orientation. They can give orientation on some degrees-of-freedom and with more or less accuracy. A mix of them and software can give very accurate results with up to 6 degrees-of-freedom (20). Those are called inertial sensors or systems.

Gyroscopes are able to sense rotation in one to three degrees of freedom (21). This allows the robot to know its current orientation based on the last known orientation and taking into account the rotation it just undergone. Though pretty useful, gyroscopes tend to drift with increasing error as the sensor is turned. Also, a gyroscope can only detect rotation. This means it cannot give the absolute angle but a relative angle or an angular speed.

Accelerometers, as the name implies, measure (linear) acceleration (22). As with the gyroscope it can detect up to three degrees-of-freedom. It is usefully in detecting movement or gestures. But, as stated before, it detects acceleration and not position. This means that if there is a need to know current positions, a double integration has to be made (from acceleration to speed and then to position) and a starting position has to be known. This leads to drift of the values increasing the error.

Since gravity has a constant known acceleration (approximately 9.8 m/s^2 on Earth), accelerometers can also be used to detect current orientation, granted that the object is at rest. This can be used to correct a gyroscope, increasing its accuracy. However accelerometers can only detect orientation in two axis, since yaw does not result in any (linear) acceleration.

Magnetometers, as the name implies, detect magnetic fields (23). If there is not any strong magnetic field around the magnetometer, it can be used as an orientation device, sensing the Earth's magnetic field. Again, it can have up to three degrees-of-freedom. The extra degrees allow the magnetometer to always point North even if it is rotated.

Since the Earth's magnetic field is coplanar with the only accelerometer's axis that can't detect rotation, the magnetometer can be used to correct the third axis from on a gyroscope.

1.3.3.3 - Visualization

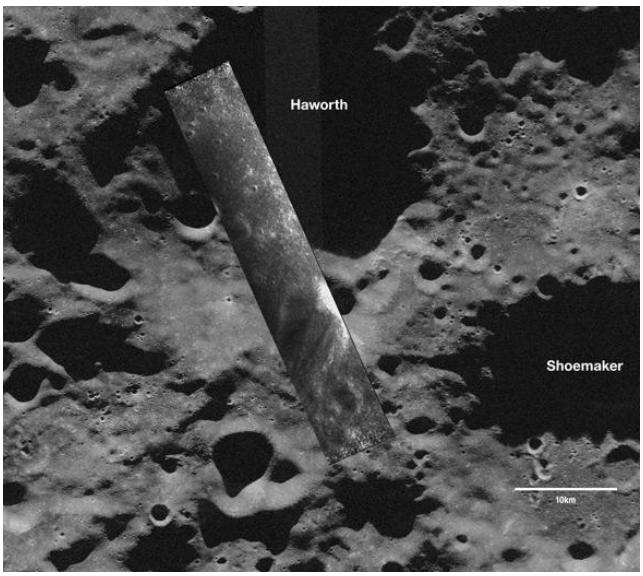
Cameras are pretty much like our eyes: they capture light and transform it into information. They can be used to directly feed other devices that will processed or output this information. Before this information is outputted, it can be filtered to display only what's relevant or it can simply be processed and never displayed as it was originally captured.

Cameras can be used as a way to bring back visual information to users that are far away or can be used to extend human's capabilities. Examples of such extensions are infra-red cameras to enable night vision or thermal cameras used on Search and Rescue missions. Another example, but this time of a very common feature, is the zoom function which magnifies far objects.

Since humans have stereo vision, which grants them a sense of depth, two cameras can focus at the same point from a slightly displaced position and each one fed to one eye.

Cameras can be robotized to allow for control of where it is pointing at. It can be manual which can require a complex set of controls for every function it supports or it can automatically follow objects of interest or, at least, have some automation (automatic zoom, focus or aperture size, for instance).

Images do not always come from cameras. As examples, radars can create images with the detected objects or a moving laser can create a three-dimensional image.



This image is an example of an image that was created by a source other than a camera. In this case, it was created by two radars, one on the Moon's orbit and the Arecibo Observatory radar.

Img. 5 - Radar imaging of the moon (80)

1.3.3.4 – Other Sensors

There are many more sensors that were not covered on this sub-chapter. Some may have more utility for teleoperation than others but, if the task requires, virtually any sensor may end up being used to this objective.

1.3.4 – Information Delivery

The humans have a range of ways of gathering information from the environment. One can compare it to the sensors from the previous section. Since the human is not where the robot is, sensors capture information about the environment and this information is then delivered to the human. Processing can be done to the information before presenting it to the human, for instance, to reduce noise or to derive some other information (the GPS does not give the raw data to the human – it gives the geo-location and some other relevant data).

Sound

As stated on *Human Factors Engineering* (8), "(…) the stimulus for hearing is sound, a vibration (…) of air molecules". We use sound as an information carrier. It tells us about the world around us and warns us for incoming dangers. Information about objects around the world can be extracted using sound alone. For instance, we can know the position of and object or if it is moving, its direction and speed. And all this is achieved without shifting our attention because sound is omnidirectional. Communication can also be achieved through the use of sound. Humans developed spoken language which uses sound as the medium for information transmission between individuals.

Alerts/Alarms – Sound is a good way to warn the users because, as stated above, sound is omnidirectional (8). Alerts can be given without distracting the user from his task. This not only helps achieving a faster feedback as it adds one channel of information to the user. But over usage of alarms can have the opposite effect. When too loud, too many or too frequent alarms are used, the user will end up trying to shut down the alarms instead of acting to solve the problem that caused the alarm on the first place. An example is on a flight cabin, when the airplane is stalling, there is so much noise and lights flashing that the pilots tend to try to shut the alarms off before correcting the airplane (8).

Information – Sound can be used as a medium to carry information. Raw sound, by itself, already carries information (for instance the position of what's making the sound) but this information might not be what is needed on a system. Sound can vary, for instance, in pitch and repetitions to achieve a message. Rhythms, variations and sequences can also be used for more complex sounds (8). Sound can be directed mapped to something else (as an example, the hospital's heart monitor that sound like "beep, beep, silence", representing the heart beating; the medics can extract several conditions out of those patterns) or can be abstract (such as a fire alarm). More complex sound can be used to simulate the real world (in a simulation) or even to transmit the current state in a distant place (teleconference or teleoperation). Again, care needs to be taken not to overflow the user with sound or it will become noise and will damage more than will help.

Synthetic Voice – Artificial sounds can only go so far. A way to increase information delivery is to use something we already use on an every-day basis: verbal communication. Synthetic voice can be used to deliver more complex information using information that does not require the user to learn and remember arbitrary sounds and patters (8). Recovering the airplane example, instead of using "beeps" to warn the pilot of a problem, the computer can say one or two words that identify the problem right away (for instance "Stall").

Surround – As already stated, sound us omnidirectional and the information we can get from sound is, for instance, position or speed. But to extract that kind of information, more complex systems need to be developed. Surround sound systems use multiple sound sources and special hardware/software to create a feel of three dimensional sounds. This helps to create the sense of having moving and tri-dimensionally placed sound sources. Surround can be applied in teleoperated robots for search and rescue teams. The direction of a sound can be identified (for instance knocking), allowing the rescue team to find survivors (12).

Vision

As stated on *Human Factors Engineering* (8) "Essentially all visual stimuli that the human can perceive may be described as a wave of electromagnetic energy". Vision carries a lot of information of many kinds. The human visual system can sense brightness, color, contrast, depth, movement (between others). And the human brain can expand these capabilities by adding, for instance, patter and shapes recognition (8) (24).

Color can deliver good information but some people are color-blind. Using only colors on an interface to show information will create trouble for those that cannot distinguish between colors. Contrasts and patters can be used instead (traffic signal use the position besides the color (8)).

When in poor light, our color vision is lost. But we can still see. This is due to a wide range of light conditions under which our eyes can accommodate to see.

Contrast, as well as shapes, is a good way to quickly distinguish between two objects. It can be used to separate two objects on a visual interface, while not suffering from the issue of colors with color-blind people.

Movement is a good way to create visual alarms. Humans are sensitive to movement on the sides of the field of view (slightly away from where the eye is looking at). This sensitivity can be used to call the attentions of the users. But as with the sound alarms, abusing visual alarms will annoy the user and disturb his work.

Simple Devices - There are a number of different simple devices that can output information. L.E.D.s (25) can inform of some binary status or can alert the user to some event that is either occurring or not (for instance an alarm). L.E.D.s can also form arrays to display alpha-numeric data. For a little more complex and continuous type information, analog meters or gauges can be used. Those displays can show simple information in a set range but are limited to that range. Still they are very useful as a cheap low end solution if the range is expected to be known and not variable.

Displays/Screens are capable of showing multiple, custom information at the same time, depending on its resolution (the number of points, or pixels, it can show). It is, probably, the current most important information output device. It works as a good evolution of simpler devices that are capable of displaying visual information. Another advantage is that if the user needs something that is not displayed at the moment that information can switch with the one that is current on screen or both can be showed at the same time using free space or overlapping. This allows for, virtually, an infinitude of information be showed on a single display as opposed with a simple L.E.D. or a meter or gauge.

Heads-up Displays (HUDs) are a way of showing complex information in a small space (12). They take advantage of displays resolution to show more information than it would be possible with L.E.D.s and other simpler visual devices. This is not a physical device like the other examples but a discipline of placing and displaying information on screens.

Stereoscopic Vision is a way of perceiving depth using two slightly different images (26). This third dimension is achieved by two, slightly apart images focusing on the same spot. Then the images are processed by the brain and the differences make up the depth information. This method isn't as accurate as direct distance measurement but is one that our body uses in an everyday basis. On the sensors sub-chapter, it is explained how to create these images using cameras placed slightly apart.

Head-mounted Displays are devices that are used on the head. They can take the images created by the stereo camera and create stereo vision by having a screen showing one image to each eye (27). These devices can have sensors installed to give it more functionality but it will be discussed on proper place under the next sub-chapter *Controllers*.

Haptic

As stated on *Human Factors and Ergonomics* (24) "The somatic sensory system is composed of four distinct modalities. Touch is the sensation elicited by mechanical stimulation of the skin; Proprioception is the sensation elicited by mechanical displacements of the muscles and joints; Pain is elicited by stimuli of sufficient intensity to damage tissue; and Thermal Sensations are elicited by cool and warm stimuli". That describes the four kinds of sensations present of our somatic sensory system which can be exploited to interface computers with humans. It is a sense that is present on the whole body and allows us, not only to know our own body's positions as well as the physical world and objects that we interact with. This can be used to control remote and precise devices. For instance, remote surgeries can use haptic feedback to help the surgeon to feel what he is doing (28). If he doesn't feel the pressure he is applying when he is making a cut he may damage tissue. If that tissue is part of some vital organ he might kill the patient. Another example is the usage of haptic feedback to improve virtual realities (29). The user can feel what he is interacting with giving a more believable simulation.

Vibration is the simplest way of (interactive) haptic feedback. In its simpler form it is pretty much the like a L.E.D.: it warns the user of an event, but on a more discrete way. But it can also transmit more complex messages using the frequency, timing and intensity of the vibration (30) (just like sound alarms). Some more complex effects can be achieved with this. For instance: the bumpiness of a road can be simulated on a wheel joystick using vibration. Another way of using vibration as a haptic feedback is the called electro-vibration (31). It creates a sense of friction using electrostatic vibration. This sense could be used, as the makers say, to feel the weight of a folder the user is moving, representing how many files it is inside or to make keyboards on touch interfaces to be felt.

Force-Feedback enables the user to receive the reaction to his action or to something that is acting upon what the user is controlling (32). Simpler devices may be designated as force-feedback devices but only deliver vibrations. But most force-feedback devices do deliver forces that represent a real or virtual world reaction that is directly mapped what the controller represents. For

instance, steering wheels (a special kind of joysticks that simulate driving a car) may enable the user to feel the road bumpiness using vibration. This is acceptable for simple vibration-like feelings, but to simulate the centering of the steering wheel caused by the car's forward movement or a rock hitting just one front wheel a force is needed to be applied on the device. A less ludic example is remote surgery. The surgeon has to know how much force he is applying to the tissues or he may damage them (28). So, the device transmits these forces to the surgeon using force-feedback to make the surgeon feel what he is doing.

Texture is a feature inherent to the objects we touch. It can help identify objects or the state of them and can help giving a deeper immersion on a simulation or teleoperation. It can be simulated using, for instance, gloves with actuators on the finger tips and hand palms (33) or using electromagnets that act on a magnetorheological fluid (a special kind of fluid that reacts to magnetic fields), creating (possibly dynamic) textures on a touch screen.

1.4 – Controllers

The intentions of a user before a system require a way to enter data into it. The entering of this data into the system is called input and is achieved using an input device or controller (34). This input device should be chosen taking into account the task to be achieved and the device's performance on the task. The tasks should be completed effectively and efficiently in an easy and straightforward fashion. But the users that will perform the task should also be taken in account. Their physical and psychological characteristics may require a different kind of device to accommodate their limitations.

There are a number of different controllers on the market and even more are custom-made or are in prototype stage. All of them have been designed to accomplish a set functionalities but the user and developers are the ones that dictates what a controller can do. An example is the mouse as a pointing device. Despite this fact mice can be used to control cameras' direction.

1.4.1 – Discrete vs. Continuous

Numerical data can either be discrete or continuous. Discrete data is an interval that has a finite number of divisions (such as the number of coins on a pocket), while continuous data is an interval with infinite number of possible divisions (such as the distance from Earth to the Moon) (35).

A controller can also be of discrete nature (also called digital) or of continuous nature (also called analog). Digital controls have two or more finite positions (for instance a key of a keyboard can either be pushed or released) while continuous have a virtually infinite number of positions (for instance a joystick) (36). Each type of control has its function. Digital input is usually used when two states are needed (for instance on and off). A three or more states can also be of digital nature (for instance the gears of a car). But for continuous input (such as a direction or the amount of power) analog inputs are usually used. A digital input can have so much resolution that it simulates an analog input. Actually, information is ultimately processed as discrete (binary code) but the resolution to use can be chosen.

1.4.2 – Controller Types

Following is a set of possible controllers that can be used on a teleoperation. Those are the readily available on the market. But sometimes a custom controller must be built to suit the needs of a certain task.

Mouse and Keyboard

The mouse and the keyboard are the most common set of controllers on personal computers. They have different functionalities and cooperate on the everyday work on a personal computer.

The keyboard is used as an input device to enter alpha-numeric information in a computer (34). It is a discrete controller so it can also be used as a way to input discrete information (besides the alpha-numeric data). If correct mapping is made, the keyboard can simulate a continuous controller by using time to map a continuous action (example: the longer you leave the finger on the key the more throttle you will apply on a car).

The mouse is, by definition, a pointing device (34). Using a cursor, it points and interacts with anything on screen, pretty much as if it was an extension of one's hand. But its analog movement can be used to control other things, for instance, the camera on a three-dimensional game, more specifically, its Yaw and its Pitch. It can only control those two proprieties of the camera due to it only having two axis (unless the scroll button is used for a more rough control). There are 3D mice in the market (such as those from 3DConnexion (37)) that add a third degree of freedom. 3D mice (see (38) (37) for examples) are not as common as normal (2D) mice. They are more used by design professional. This comes from its ability to perform several navigation tasks without having to change the mouse mode: the user can pan, zoom and rotate continuously using a 3D mouse. With a 2D mouse the user has to select pan and then the two dimensions of the mouse are mapped to the panning action. Then, if the user wants to do another action, he has to change it to zoom or rotate so the mouse will act that way.

There are other devices that are not mice but can simulate the function of mice (or more correctly, pointing devices), even the new third dimension. Between others examples are the wiimote and the falcon (39).

Img. 6 - 3D Mouse and some of its possible actions (38)

Joysticks

Joysticks can be used as a game controller or as a robot controller (34). Joysticks are usually used when there is a need to input both direction and speed. Most joysticks are hand-held and ergonomically designed so they can be comfortably used. Also they have less buttons than a keyboard which are

displaced, again, ergonomically and on an easy to remember fashion. Another characteristic, not present in all joysticks, is one or more analog input. Since many world interactions are in a continuous range and not discrete, this makes analog a better solution than digital. Though a keyboard and a mouse is more adequate for every day work (but not every kind of work - 3D designers may benefit from using a 3D mouse or a touch device), gaming and teleoperation are best served by some kind of joysticks, probably, merged with some other controllers for added interaction.

Gamepads are another subgenre of joysticks. They are usually held by both hands and have a D-Pad and/or one or more thumb-sticks (a smaller version of a simple joystick that is to be used by thumbs).

Steering Wheels

Steering Wheels are another sub-set of joysticks. They are designed to simulate driving a car. This means that they have a wheel and, optionally, two or three pedals and a gearbox. Usually all of these controls are analog allowing for a realistic simulation. This controller type can also be used to control robots that are fashioned or behave like a car. If the robot has more functionality, the driving can be accomplished using a steering wheel while other functionalities are addressed by other controllers.

Touch Screens

Touch Screens are, currently, raise on the market, where most recent mobile devices have one. They allow the user to directly select or control objects on screen. This makes the screen both an input and an output device (34). Haptics are also being applied to touch screens to add yet another output/feedback capability (besides the visual) (31).

Keyboards can be simulated by drawing it on screen and then the user can type directly into the screen. The haptic feedback of a keyboard can also be simulated using, for instance, the TeslaTouch technology. Still, the haptic technology is still not of common use on the market being this is a down point: the user does not have a feel of touch when using the touch screen (it is hard to type without looking while you can easily do it on a physical keyboard).

Another downside is that if you try to compensate for physical devices by simulating them on screen, you will lose space on the screen. Also, the cost is higher than a normal screen plus the keyboard.

Gesture Recognition – Hand-write and gesture recognition helps humans to input data in a natural way (34). The user can, easily, write down using normal handwriting or can make gesture to control what the computer is doing. Some gestures feel more natural than the common computer input. For instance, the user feels tempted to drag an image with his hands. Then he may throw the image to make it move to where he wants or may try to zoom in by stretching the image with his hands. All this feels natural opposed to the common mouse "selected new action and then act". It started with gestures using a pen (40) and now uses both touch screens (41) and camera-based technologies (42).

Kevin Arthur gives a few points on gesture design (43). When designing gestures, care should be taken not to use unnatural gestures. For the designer it may feel natural because of training and knowledge but not for the user. Good clues should be given and good feedback should also be given to the user's actions. Also the space should be enough (an issue with small touch screens) for the gesture to be comfortable.

Head-Mounted Displays

Head-Mounted Displays were reviewed before but as a way to output information. But they can also be used as a controller. Sensors can be used (for instance gyroscopes and accelerometers) to extract head position and movements (12). This information can be then used to control a camera, for example. It then becomes a natural way to look into a virtual world (or a physical remote world through teleoperation).

Voice Recognition

Voice recognition is not a physical input device. It is a way of inputting information using voice capture devices, as microphones, to detect the voice and then analyze it (34) (44). Algorithms then try to extract words from it and they make up whole sentences or instructions. But there is still a lot of room for

improvement since those systems are error prone. This comes from the difficulty to program systems that can deal with the various accents and even individual variability in the way a word is spoken. Those problems can be mitigated by training of the system to a person's voice.

It is better used on systems where "eyes-free" are needed (44). Those systems are used in situations where the user cannot or does not want to look at the device (for instance, while driving). It can also be used by disabled people that cannot use normal input devices to interface with computers and other devices.

Other Controllers

There are a number of other kinds of controllers. The following three are just examples of common mechanical controllers that can be used to build more complex ones.

Switches – The Switch is a key or a button – it may be on or off / pressed or released. As so, it is used to control discreet things that can either be in one of two positions. This also limits its uses to a single aspect being controlled or, if multiple aspects have to be controlled using the same switch, a more complex interface has to be built that can change controllers' functionality.

Levers – A Lever is an analog controller. It has "infinite" positions (it has a resolution that limits the number of positions and a way to transmit the data the further limits it), unlike a switch. It can be used to map, directly, the position of an object to the position of the lever; for instance an airplane's flaps. Or it can be used to adjust some value between a range of values, for instance speed or acceleration.

Rotary Controllers are like levers but work on a rotary fashion. They can work between a physical range or rotate freely. Just like the levers, they also can be used to control values between a range and to map objects. But since it rotates, objects that also rotate are best mapped by rotary controllers. Rotary controllers can also be used instead of levers if the space is too crowded for a lever as a small rotation is more comfortable than moving some linear switches too far away.

1.5 – Teleoperation Issues

There are a number of issues that can occur during a teleoperation session. Some of those issues are reviewed on this sub-chapter.

1.5.1 – Latency

"Latency", generally speaking, "is a time delay between the moment something is initiated, and the moment one of its effects begins or becomes detectable (45). A common example is when someone tries to visit a Web-Page. The user enters the URL and then there's a (usually) small delay before the Web-Page starts showing. A more serious example is when a surgeon moves his arm during a tele-surgery. There is a delay from the time he moves to the time the robot arm moves to the time he sees it moving.

1.5.1.1 – Why does it exist and its consequences

Latency, specifically on engineering, has three sources (46). Information has to travel using mediums and those mediums have a maximum propagation speed. This means that the information takes time to get from one point to another. Another is the processing that information has to go through. The sending and receiving computer have to process the information as do the routers and other equipment on the network. And finally, there is the traffic that the transmission medium may have to endure. There are queues of packets on the routers that need to be processed. This happens because the information needs to share the mediums and devices of the network. If the queue is bigger than a maximum value, packets may even be dropped (increasing the time delay even more as the packet needs to be resent).

Travel Speed on a Medium – The speed that a piece of information can travel on a medium usually is in the range from 70% up to 95% of the speed of light on vacuum. While this might not seem like the cause of a major delay, if we try to communicate with somewhere outside the planet Earth (for instance, space missions) then we start to notice some major delays.

Elaborating a little more the space example, let's suppose we're controlling a robot on the face of Mars. On worst case scenario, let's suppose

that Mars is in conjunction with Earth (the planet's positions on the Orbits put then on opposite sides, in such a way that they are at the maximum distance from each other). This puts 401.3 million km (47) between them. Since there is mostly vacuum from Earth to Mars, besides the Atmospheres, let's suppose that the radio-waves we're using to communicate with the robot travel at 90% of the speed of light. This means that the radio-waves travel at 269 813 212.2 m/s. So, a round trip from here to Mars and until we receive the reply takes, approximately 2 975 seconds, or 50 minutes. This raises big issues for long-range teleoperations as the operator must way a long time to see the consequences of his actions.

Processing Time – Another factor is that information has to be processed. That processing takes time, adding to the delay. So, the more points (for instance, routers on a network) a piece of information has to pass by, the longer it will take to arrive. Also the information has to be processed, sometimes heavily, before it is presented (decompression, rendering as a couple of examples). This problem can be mitigated by brute force using just the increasing processing power available. But good algorithms and planning can also aid.

Traffic – If a transmission medium is shared it will have traffic. And this usually happens because a good way to reduce a network's expenses is to share it. Sharing a resource means that, at least, two users will try to use the same resource at the same time. With thousands or even millions of users, a network can get congested, causing delays (and information may even be drooped).

Usually, there's no congestion on robotics because the channels are dedicated. But more complex problems that require networks of robots and controllers might have this problem.

1.5.2 – Bandwidth

Bandwidth is "(...) the amount of information that can be passed through a communications channel in a given amount of time; that is, the capacity of the channel" (48). If a lot of information has to be transmitted it can be transmitted during a long interval time with low bandwidth or during a quick interval of

time using high bandwidth. Bandwidth can be an issue for long range teleoperation which already suffer from high latency. For close range communications, it can counteract latency (if the data can be sent in parallel or faster than the latency).

1.5.3 – Run-time Robot Problems

There are problems that did not exist or were not identified before the teleoperation session. Those issues are reviewed on this section.

Software/Hardware failure – software problems are quite common. It is inherent from the complexity and lack of malleability that software development has. Some problems can only appear when a number of set variables get to a state (which can never occur). Some situations are so rare that are not predicted and, when they do occur, the system cannot handle them. As well as software, hardware can have design problems that are not identified in time. But, contrary to the software, hardware suffers from physical issues such as heating, damage and wear. Most software failures can be recovered from if predicted but hardware failures are harder to overcome. One solution is to have a redundant system or part of it to replace the broken one.

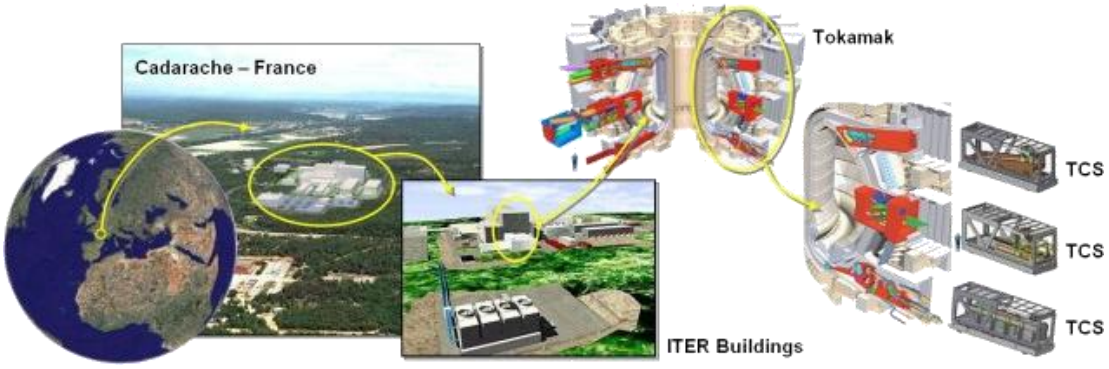
Autonomy during communication silence – Robots working on some conditions may lose communication with the operator. For instance, robots working on the far side of the moon or Search and Rescue robots that went too far into a building. If the robot cannot communicate back with its operator, it means it has to have some autonomy. If the robot has no autonomy, it may be lost or will be useless while the communications are not restored. As an example, a Search and Rescue robot should be able to continue searching on its own if it cannot communicate with the operator. Or it should come back to a position where it can communicate.

2 – The ITER project

2.1 – The Project

The ITER (49) (50) (International Thermonuclear Experimental Reactor) is a multination research and development megaproject composed of the European Union + Switzerland, Japan, United States of America, Russia, China, South Korea and India (51). It has a massive budget that is already on the 16 billion euros (52). It is a long term project: the reactor is supposed to be completely assembled by around 2018 and, one year later, the first plasma generation. Energy is expected to get to the grid around 2040 (49).

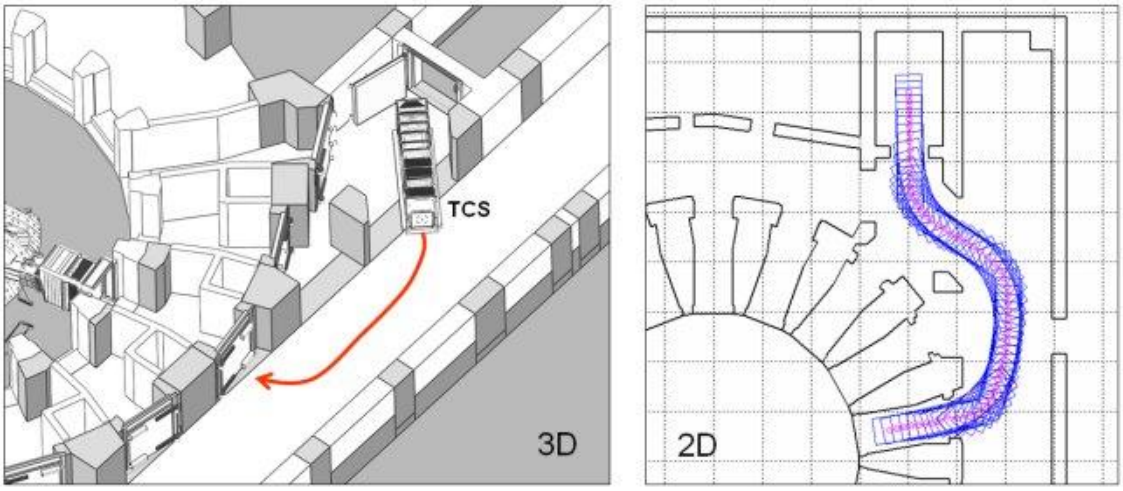
ITER's aim is the study and production of fusion power. This kind of power is intended to be used in electricity generation as a more efficient and safer method. Fusion power (49) (53) generation is safer than fission and pollutes less than coal and fission. It has no risk of a meltdown event because the reaction cannot sustain itself and the amount of fuel used at any times is substantially lower than on a fission reactor. Still, environmental accidents and radiation exposure can happen since the reactor is radioactive and produces one radioactive isotope (Tritium). Another advantage of this kind of energy generation is its output. It can output four million times more energy than the current chemical reactions from burning fossil fuels. This leads to a very low usage of fuel for the same energy output. According to iter.org, a 1 GW fusion plant would use only 250 Kg of fuel per year, while a coal counterpart would use 2.7 billion Kg of coal. And to add to that, it uses a very common element as fuel: Deuterium, a Hydrogen isotope and Helium is the main by-product of the fusion.



2.2 – The Reactor Building

To study fusion power, a reactor is to be built in the south of France. This reactor will be built on a complex, more specifically in the Tokamak building. Another building on the complex, the Hot Cell Building, will be providing the maintenance infrastructure. The reactor has three levels, accessible from a service elevator.

The reactor building is the main set for this thesis. Since it contains the reactor itself, it is a very dangerous place for a human being to be at. The reactor is radioactive and implements other dangerous technologies such as cryogenics and strong magnetic fields. Its tight and dangerous corridors are not to be traveled by humans but by a vehicle presented on the following section.



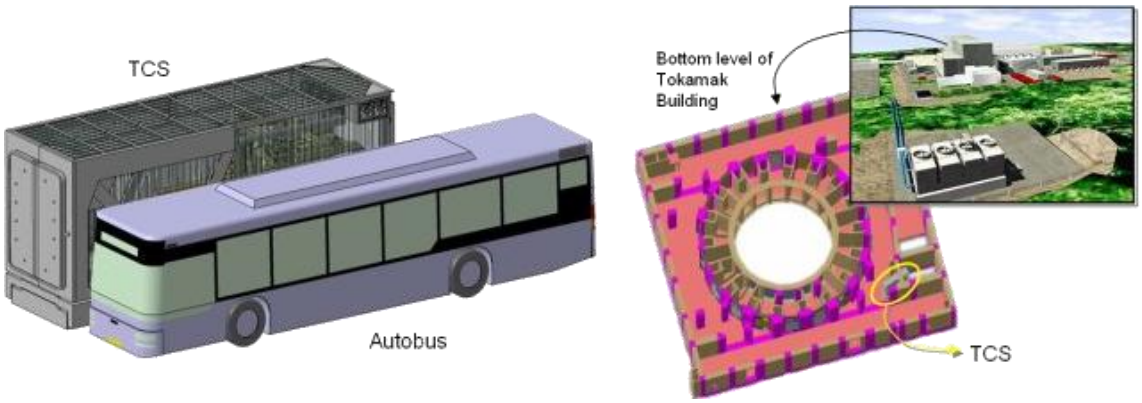
Img. 7 - The Reactor

2.3 – The Hovercraft

Despite the fact that the reactor is safer than a fission reactor it is still not a safe place to be, so, no human is allowed inside. A Transfer Cask System (TCS) is required to transport fuel and other materials to and from the reactor. The TCS is an autonomous 100 Tons vehicle that slides over an air cushion. Its size, similar to that of a bus, and its weight (translates into inertia) makes it hard to drive inside the tight reactor level. The small maneuver space and the fact that it moves nearly 100 Tons on a near zero friction environment make it a big challenge, even for a computer, to drive. Adding to this complexity, but at

the same time, to its maneuverability, the TCS is expected to have two or four wheels with 360 degrees turning capabilities, aligned on the long axis. Those wheels are responsible for the traction and guidance of the TCS. To mitigate the driving difficulty and to reduce the risk of collisions, the TCS should move extremely slow, predicted to be from 1 to 10 cm/sec (around 0.0036 km/h to 0.036 km/h). But, other vehicles are expected be moving inside the reactor building at the same time, adding to the already difficult driving.

To allow automatic (or manual but remote) maneuverability, the TCS is to have an array of sensors, range laser finders and cameras and possibly other kinds of sensors. But those sensors or even the communication can fail or suffer interference. Solutions for those failures are still under research. One solution is to fall back to a manual control scheme. But manually controlling this complex massive machine inside the tight reactor building is a difficult task. Finding a good and intuitive way to achieve this task is the objective of this thesis.



Img. 8 - The Transfer Cask System (TCS)

3 – System Architecture

3.1 – Introduction

With today's computer processing power we can easily achieve some astoundingly graphical pieces of art coupled with the joy of believable real-time physical interaction and all this on a modest piece of hardware.

The ITER project has the need for a hovercraft simulation with both a graphical and a physical component. None of these components needs to be top notch but they do require running smoothly on any mid-range computer to be useful. Still, it does not have to run on almost every single computer out there so there is no need for multi-platforming.

This chapter will focus on the choices made to build up the simulation's architecture based on the existing hardware and software.

3.2 – Platforms and Operating Systems

Hardware and Software platforms, including Operating Systems are the base upon which software systems are built. For some systems this may be a very important decision, and sometimes, a vital one. For others, it is just as trivial as choosing the most convenient one.

3.2.1 – Platforms

Today's most common end-user platforms are the PC, gaming platforms and the growing market of mobile devices.

Both PC and gaming platforms have plenty of processing power. They have multiple dedicated processors (e.g.: sound, graphics and physics) and usually multiple and/or multi-cored general processors. The mobile devices are growing their processing capabilities but they focus on mobility/autonomy and are yet no match for more powerful platforms.

The control devices also change with the platform. The PC has the widest choice of controllers. It can basically work with any controller, be it commercial

or custom made (some may not be able to be used or even plugged without some work). But this platform is known for the keyboard and the mouse. Gaming consoles have controllers oriented to the platform's objective: playing. Some can even accept the keyboard and the mouse.

For this project, the common PC platform will be good enough for its ease of use and broad support. It is also the one that best supports the usage of custom made hardware, which is one of the requirements.

3.2.2 – Operating Systems

Since the PC is the platform of choice, only PC Operating Systems need to be review. On this platform we can usually find Microsoft's Windows, Apples' MacOS and UNIX based systems.

Despite the fact that all three are mature enough, Microsoft Windows was chosen due to the existence of the tools and support required on this project.

3.3 – Rendering

Rendering is an important part of any (visual orientated) simulation. It allows users to see the virtual world and act upon it. Also, the user can see what effects his action will create upon the virtual world mirroring what would happen on the real world.

3.3.1 – Software

There are a lot of graphics libraries that provide some kind of graphical features. These range from OS's UI APIs to full 3D engines, with some specializing on the 2D.

The two most used 3D graphic libraries on the PC platform are OpenGL (54) and DirectX (55). While SDL (56) is still good library, its capability are only in the 2D field and so will be discarded as a choice. Despite being considered on the intro, OS's UI APIs just do not have enough power to be of any use on this kind of simulation.

Ogre (57) is much more than rendering software. It is considered (by its makers) an "Open-Source 3D graphic engine". It is built upon DirectX and OpenGL for rendering and other APIs for other kind of work (for instance PhysX for physical simulation). It is also multi-platform.

OpenGL and DirectX are direct competitors. One can say that they do, basically, the same. They allow developers to render 3D scenes in real time while taking advantage of hardware acceleration. The main difference is that OpenGL is open-source and multi-platform while DirectX is proprietary and present only on Microsoft platforms (on PC but also on gaming platforms). Despite DirectX being proprietary it is free just like OpenGL. The performance is now identical for both despite OpenGL's best performance when it was first released. Hardware support is also the same for both: major manufactures always fully support DirectX and OpenGL on their new hardware and drivers releases.

Since they are so close, DirectX ended up being chosen by a simple question of taste. The simulation could have been built using OpenGL and achieving the same results.

3.3.2 – Hardware

It has been a long time since the graphics had to be rendered on the CPU. The GPU came to the aid of gamers allowing for faster rendering times and, consequently, better frame rates - or more content on screen. This ability to have good graphical components on the games attracted more gamers which, in turn, attracted more developers, creating an endless loop. Following the industry the GPU development "exploded" and now we're moving to an ever closer movie quality real time games.

The two main GPU makers are NVIDIA (58) and ATI (59). Both of them are in constant development of new hardware and the quality, performance and price tend to accompany that of each other.

3.4 – Physical Simulation

The physical component of the simulation is responsible for calculating how the world should behave, physically, to the user's actions or any other kind of physical interaction. The technology is evolving in such a way to deliver faster and more realistic physical simulations through software, hardware or even a mixture of both.

The current physical simulation software on the market enables it to be run on almost every computer. For instance, NVidia's PhysX can run on any GPU of theirs, starting on GeForce 8 (60) (six iterations ago at the moment of this thesis writing) with the ability of falling back to CPU support in the absence of the required hardware. This fact, coupled with the fact that the software to build those simulations is being distributed free, allows for very accurate simulations to be developed easily and without reimplementation of the low-level physical models of the world.

3.4.1 – Different Kinds of Simulation Software

Nowadays there are two kind of physical simulations software. There is the scientific and game simulations software.

The first type is more accurate and, often, problem specific. The increased accuracy means that there's a need for more processing power or to make the calculations offline. It is used, for instance, in artillery ballistics simulations or cars and airplanes friction/aerodynamics models.

On the other side, game simulation software is aimed at general applications, usually real time. One can build most simulations with it with a penalty on accuracy. This reduced accuracy comes with the need to maintain a good frame rate while simulating hundreds to thousands of simultaneous objects. Even with less accuracy than its scientific counterparts, gamers are usually satisfied with game's physical simulations accuracy. They're even used on offline simulations on 3D design programs (like 3D Studio Max (61)) with good and believable results (62).

Under the scope of this project there's no need to review scientific physical libraries, only the game ones.

3.4.2 – Software

Both PhysX (63) and Havok (64) are high end and real time physical simulation engine. Both are proprietary and are maintained by two giant competitors. NVIDIA (58) bought PhysX from Ageia and now maintains it. NVIDIA also developed support for PhysX on their own hardware (on GPU), with back compatibility down to GeForce 8. ATI owns Havok (more correctly, AMD (59) owns it) but its hardware gives support to Bullet (65). PhysX has a whole physic related features in one free package while Havok is registered (was paid) by title/feature pack and is business orientated. Both offer paid technical support.

Newton (66) is the most scientifically accurate physic library of the reviewed libraries but also the slowest. While it may be better when there's a need for accuracy it will fail when frame rate is vital.

Both ODE (67) and Bullet are open source. ODE is the 3rd most used library (PhysX is the first followed by Havok (68)) but Bullet might get closer since it's supported by ATI's hardware and ODE is currently not under development.

3.4.3 – Hardware

With the demand and development of accurate real time physical simulations grows the need to build specific hardware. This frees the CPU load, allowing it to be used on other calculations. The first dedicated PPU (Physics Processing Unit) was built in 2006 by Ageia (previously known as NovodeX and then bought by NVIDIA). The PPU is called PhysX, also the name of the current software physical library of the same company. PPU never was widely adopted and is being replaced by General Purpose Computation running on current GPUs (Graphic Processing Unit). This means that most of the people with a relatively recent GPU (GeForce 8 (60) and above for NVIDIA and x1XXX (69) for ATI) can use this hardware to run physics. This not only gives the frame rate a boost but also frees CPU resources. The physics gains a boost from the GPU due to its better parallel performance, as stated in the above Rendering

section. This is true because a physical virtual world is composed of hundreds or even thousands of interacting objects. The math in floating point is also easier than in integer, another spot where the GPU surpasses the CPU.

3.5 – ITER Hovercraft Simulation Architecture

The simulation runs on a PC with Microsoft Windows OS installed. This platform enables ease of development and testing (same machine for development and testing / no emulator or transference needed) and provides enough power to run a smooth and accurate enough simulation. The large array of tools present on the chosen OS also aids on the development. And since it is the most used worldwide, the simulation can be used virtually anywhere.

DirectX is one of the best graphical libraries and works only on Microsoft Windows (if the choice of the hardware platform is a PC). This is coupled with the fact that its equivalent ``one of the best'' physical libraries, PhysX, is also present only on PC/Microsoft Windows. The physical component should run at a constant 60 cycles per second to achieve maximum accuracy (stated on PhysX documentation). This means that a separate thread for the physical calculations is desired.

The above architecture provides the common user that owns a mid-range computer a smooth frame rate (24 being the minimum desired frame rate - the human eye/brain frame-rate perception) with enough physical accuracy for the simulation to be believable.

4 – The simulation

4.1 – Introduction

One of the big responsibilities of this thesis is the simulation. The hovercraft in question is a prototype which is still on engineering process and no real world version (to the public's knowledge and to the date of this thesis) is yet built. Also, even if there was a real hovercraft, testing with it would be impractical due to financial costs. That's why this simulation is built and then used to test different controls prototypes that could be directly used and perfected to control the real hovercraft.

This simulation required a considerable amount of time and effort to build. It was the block of this thesis that took longer and harder to build. Its development started near the very start of the thesis' state-of-the-art research and ended just as the user-tests started.

Coupled with the simulation itself, a graphical and physical (software) engine was also developed. This engine keeps the simulation clean and easy to update and correct, leaving repetitive tasks, as well as initializations and clean-up tasks to the engine itself.

4.2 – Responsibility Separation

There are two main responsibilities on the simulation. There is the simulation itself and the software responsible for the communicating with the hardware (the controls). These two are separated in two different programs. This allows for easier development and debugging as well as to simulate the real world issues of a remote controlled hovercraft. One can have the controller on one computer and make its commands pass through a network to test it for robustness. Or one can test lag and signal loss (two very possible problems on the real scenario due to the radiation) putting a filter between the two programs.

Since the controller communicates with the simulation using network communications (based on OSC protocol), one can take advantage of this and

work with it as a simple plug-in system. This allows for development of different software and physical controls layouts. One example is that, despite the fact that this was built to test hardware controllers, it can also be used to test automatic control software as it could communicate the same way.

4.3 – The Hardware handling program

It is this program's responsibility to receive the information that the hardware sends, process it and send it to the simulation. This is a very simple program (around the 150 lines of code) composed of, basically, three components and main program part (responsible for the initialization and shutdown of the hardware as well as the program cycle). It runs on a command prompt with no fancy interface. It only displays initialization and error messages but can easily display current hardware status when needed.

There is one component for each of the two hardware type (each has its own API and way of dealing with so it needs different components). And the third is responsible for sending the controller information to the simulation, using *liblo* (70), an *OSC* (71) implementation for the C language.

The *Arduino* (72) is one of the hardware components. It needs to be pooled frequently, through a serial connection to get its status. This hardware is composed of two linear (potentiometers) controllers which control the hovercraft's wheel rotational speed. It returns data from 0 to 1023. 0 represents one extremity of the control (0 volt) and 1023 represents the other (5 volt). The hovercraft is supposed to move at a speed under the 1 m/s. So, it was decided to divide the result by 1024 to get a 0 to 1 (can be interpreted as the throttle percentage) value range. Usually, this value is readjusted on the simulation since it's too slow for testing.

Phidgets is the other hardware component. It is composed of two rotary controllers which control the orientation of each wheel. They use a different approach than the *Arduino* linear controllers. A callback function is used to

process data only when new data is available (when the user rotates, at least, one controller). The data received represents its position, in units. So, it needs to be multiplied by 4.5 to get its angle in degrees (the controller has a resolution of 80 steps per 360º turn). This position can be either absolute or relative but both suffer from a problem: step-skipping. If the user moves the controls too fast it will skip steps. Since this can happen during a test session, a button that is present on the rotary controller is used to reset the position at 0 degrees.

Each cycle, the program takes the processed information on the hardware and sends it to the simulation using *liblo*. L*iblo* is a very simple API which just needs a single code-line to initialize it and another to send information (on the server side).

Arduino is very depended on the pooling frequency as the user pools its information directly while *Phidgets* use a callback. But under normal circumstances that should never be an issue as the *Arduino* is programed to update at a frequency of 60 Hz. This is also the maximum frequency that the *Arduino* can pool hardware connected to it. During development of this thesis, tests showed that higher frequency would return noisy results.

4.4 – The simulation

The simulation is the program responsible of simulating the hovercraft handling on its environment. It should also receive the data sent by the controller's input program and use it to control the hovercraft.

It is composed of three main components. It has an engine which is responsible for handling general (non-application specific), repetitive actions, such as rendering and physical simulations, on the background. Then there is the wheel controllers which are, basically, modified PIDs. Finally there is the main part of the program which initializes the program and the engine and sets up the simulation. It also is responsible to make a bridge between the simulation and the PID, which run on different threads.

4.4.1 – The Engine

A graphical and physical engine has been developed during this thesis. It is the foundation of the whole simulation, powering every *general* aspect of the simulation.

The engine is built upon *Windows API* for basic Windows OS integration, *DirectX* for graphical rendering and *PhysX* for physical simulation, between others. It integrates all those systems into one coherent and easy to use system / API. It also handles common, repetitive and non-application specific tasks (as initialing Windows API; or loading a file) needed on the simulation.

Other components were also developed. One of the other major components developed is the user-interface. This allows the user to change options and load other maps in real time.

Despite the fact that there is a program that handles the controllers, this engine also implements a controller component. But in this case it's just keyboard and mouse. These controllers are used to interact with the user interface adding functionality to it.

Also, a networking, using UDP protocol, component was also developed. This was intended to be used to transport information between the hardware handling program and the simulation but ended up being discarded. This happened due to a request from a sibling project to use *OSC* to handle communication between the two projects. So *liblo* was the first choice and which was later discarded due to a lack of C# implementations (C# is the language of choice of the referred sibling project). But since it was already in use between the hardware handler and the simulation and it was way easier than the other *OSC* implementation (*TUIO*) required by the sibling project, *liblo* stayed as the internal choice. *TUIO* was however still used as the communication handler between the simulation and its sibling project. As a note, *liblo* was never implemented directly into the engine as its code is too small, simple and specific to be on the engine.

4.4.2 – The Wheel Controller

On the simulation, the user controls the speed of each wheel, as well as its direction using the hardware controllers. But on real life we use torque to control engines at the lowest level. There are step-engines and servo which allow for direct control of its position. This direct control is not required or even needed for the normal forward/backward movement of the hovercraft but would be handy on the direction of the wheel. Still, there are limitations to *PhysX* which requires torque and forces to be applied on objects instead of directly controlling its position/orientation.

Since there is a discrepancy of what the user controls and what the engine expects to be the input there is a need to put something between the two so the right signal gets to the engine. This is where the PID comes into play.

PID – As stated above, what the user controls is not what the engine expects as an input. Basically the user sets his desired speed and the engine needs to receive a torque input to get there. As such, there needs to be something between the two receiving the user's speed output and trying to achieve it through torque. But since torque (acceleration) is a third order control whereas speed is a second order control, there is no linear way to control the engine directly. So, a PID is used to try to control the speed using torque. Very roughly, if the current speed is below the desired speed, positive torque is applied. If the current speed is above the desired speed, negative torque is applied. A constant loop ensures that the wheel speed is always closest to the desired as possible.

The direction of the wheel suffers from an even worst problem. The user controls position, which is a first order control and the engine, again, requires torque, a third order control. Again a PID is used to achieve the desired results.

The developed PID suffers from an issue inherent from the proportional nature of PIDs. It needs nearly the same time to settle, whatever the set point is. But when the user sets the hovercraft to 1 m/s he does not except it to take as long to get there as when he sets it to 10 m/s. So, a decision was made in order to improve the hovercraft's time to set point. If the set point is more than 0.5

m/s away from the current speed, the engine will produce full torque (a little less than the needed to make the wheel drift – this value was decided based on tests and not on theoretical studies). Else it will use the PID, stabilizing the speed. This will lead to maximum performance and, thus, minimal time to set point, whatever the set point.

4.4.3 – The main program component

Putting all of the simulation's components together is the responsibility of the main part of the program. It initializes the engine and feeds it the correct information so it can run correctly and also gives it processor-time every frame for it to run. It is also responsible for other initializations, loadings and cleanups of the whole application.

An important task is handled by the class *World* which is responsible for setting up the simulation's environment and the hovercraft inside it. It is also responsible for running it using the abovementioned engine.

Another task that is handled by yet another class *PID_Manager* that also falls on this part is handling the PIDs. As said above, there are two types of PIDs built on the simulation: one for speed and another direction. Each wheel has one of each type allowing the user to control the speed and direction of each wheel. What *PID_Manager* does is initializing the PIDs, feeding the user's input to them and then returning its outputs to the *World* class so it can feed them back into the physical simulation.

5 – The controller

5.1 - Introduction

Designing and building prototype controllers is one of the big, if not the biggest, issues to be completed in the course of this thesis. The main focus is to define a relatively new way of driving a prototype vehicle and build an easy and intuitive controller for it.

Since there is no real world (ITER's) hovercraft yet built and even if there was one it would be impractical to use the real vehicle, the simulation (dealt with on the previous chapter) is used to help building and testing the controller.

5.2 – The Issue

There are several issues inherent from the novel ITER's hovercraft design. Those issues are explained on this sub-chapter.

5.2.1 – When and Why

The hovercraft to be used on the ITER reactor is supposed to be automatically driven by a computer. So no direct controllers are needed to drive it. The issue is when a problem occurs that negates the possibility of an automated drive. Many problems can occur. For instance there could be a bug on the code that makes it impossible to achieve a needed task or even a physical breakdown of the hardware, for instance a sensor, which makes it impossible for a program to drive the hovercraft. Those are emergency problems that don't happen very often but really can happen. But there are also other less important issues that are prone to occur, which beneficiate from a manual control scheme. Not all the tasks can be predicted, at least not at launch. So a manual control scheme can help the operators to keep those tasks up to date while waiting for the development of new software that can handle them.

5.2.2 – The Physical Model

The ITER's hovercraft is not like a normal hovercraft. A normal hovercraft has just one point around which it turns. That point is somewhere near the front of the vehicle (the back moves sideways). As a contrast, in a common ground vehicle that point is somewhere on the back (the front moves sideways). The ITER's hovercraft mixes them both: it has two points around which the hovercraft turns. To achieve this, the ITER's hovercraft has of two wheels with a capacity to turn 360 degrees that have traction capabilities. This increases the hovercraft's cornering capabilities and all the movement agility on the very tight reactor's building's corridors, but it also increases the complexity of its driving. This is why the hovercraft is supposed to be driven by a computer and not humans. This is also why manual control is an emergency fallback.

5.2.3 – Mapping to the Human Body

Most machines that humans drive have simple designs that afford easy controls. But ITER's hovercraft does not. And there is no simple solution.

The hovercraft has two sets of wheels that turn 360 degrees and move independently. A natural choice is to map each wheel to each hand. But does that work? The hovercraft has a front and a back wheel while the human body has a left and right hand. Should we force the user to have one hand in the front and another on the back or map the front wheel to one side and the back wheel to another? Maybe there is a control scheme that can even avoid this issue all together. Sadly, no solution was found during this thesis that could avoid this issue. So, the user is given the choice to place out the front and back controls as he sees fit and feels comfortable with.

5.3 – The solution

All of the above make ITER's hovercraft a hard physical model to understand and a very hard one to drive by most people. This is why it is so important to come up with a very easy and intuitive control scheme to reduce the learning time, reduce users' mistakes and increase users' recall of how to drive it.

The following questions were asked while designing and analyzing several layout sketches. The best were then user tested.

- What should the controller control? What can it control?
- Wheel/Hovercraft Acceleration/Torque/Force;
- Wheel/Hovercraft Speed;
- Wheel/Hovercraft Angle;
- Hovercraft position;

The list of questions above includes a differentiation of wheel from hovercraft. This is due the fact that the system is more complex than the average and, controlling a status (for instance its angle instead of the two wheels separate angle) from the hovercraft, can ease its driving. But this type of control increases the complexity of the automatism needed which is what is supposed to be avoided here. Still, a compromise needs to be taken because full manual control is impossible. The hovercraft itself is remote controlled and no living being should enter the radioactive reactor's room where the hovercraft operates. So the hovercraft needs to be able to do, at least, a low level of computing. This can be used to aid on the building of its control.

It is safe to assume that the position of the hovercraft should not be directly controlled. That requires a high level of automatism which goes against the purpose of having a safe, simple fall back solution.

Speed is not a usual artifact to control and require a higher level of automatism. For instance, airplanes use auto-pilot to maintain speed and the new cars use cruise-control for the same purpose.

The most usually controlled artifacts are torque or force applied and the angle of a wheel or an equivalent. For instance, cars control torque on the traction wheels and the front wheels' direction while an airplane controls forces and torques of planar surfaces or the angle of those, if the plane is low-tech.

The most manual controller controls each wheel independently on both direction and torque (basically acceleration). This was the first (internal) test that was made and the results were disastrous. Informal tests showed that the user could not control the hovercraft. Its control was easily lost due to the complexity of controlling the two wheels' directions and made worst by the

controlling of its acceleration. The control over the hovercraft was lost as soon as any user touches the controls. One, supposedly easy task – stopping the hovercraft – proved almost impossible even if just controlling the torque of a single wheel with no directions changes at all. This lead to the first big decision:

Controlling acceleration is a bad choice. The alternative is controlling position or speed. Position seems a bad idea because it is basically a fully automatic control and can only be achieved using a complex, screen based, interface-control (the user could be required to point a position on the screen to where the hovercraft should go). Speed also needs some kind of automatism since engines are controlled by the outputted torque (or an input that should generate an output under zero-load). The needed automation receives the desired speed from the user, senses the vehicle or wheel speed adjusts the engine's torque. So, controlling speed was the choice made. This is described in more detail in the cluster entitled **The Speed Controller**.

Now that way to control the movement was decided, there is still the direction control and the controller layout to build. Since the controller is to be simple and small, preferably handhold (or at least small, even if fixed), more complex, namely screen controllers, are not considered (there is a sibling project dealing with on screen display of information to be used, possibly, in conjunction with the controller). This leaves levers, rotary controls, buttons / switches and touch technology. The last is very good for prototyping planar controllers but doesn't give any feedback, not even the feel of being touching the right control, so it's discarded. Buttons can work for atomic or double state actions (like loading/unloading cargo). But these actions aren't studied here due to the complexity of implementation. Levers and rotary controls seem to be an intuitive solution. Still, the layout of these parts plays an important part on the controller. Such was proved on the first test where a two-thumb-sticks joystick was used (basically 2 double-axis levers) and its layout proved to be a bad one.

So, several layouts were sketched on paper and where roughly analyzed. On the end, only two were selected to make the users tests. Still, there was something common on most of them: the speed was controlled using a linear controller and the direction of each wheel was controlled using a rotary controller.

5.4 – The Speed Controller

On a previous sub-chapter, it was stated that an engine outputs a certain torque from a certain input. Also, that output generates acceleration and the human controller should be worried with speed and not acceleration. So there's a need for a special controller between the physical commands of the controller and the engine. This controller is software based and transforms the user desired speed and, taking into account the current wheel speed, decides what output the engine should be having at that instant.

There's a base control algorithm PID, also referred on the sub-chapter **4.4.2 - The Wheel Controller**. This kind of controller, very succinctly, tries to achieve a set output point by correcting its input. It was, initially, thought that this kind of controller, alone, would be a very good candidate to map the user set speed to the engine's torque. The PID behaves fine if the hovercraft is very near the set speed and the PID just needs to maintain that speed. It behaves worse as the distance between the set-speed gets away from the current speed. For too far away distances, it stops being a good controller.

The engine has a maximum torque output, after which, the wheels start spinning/drifting. If the measured speed is the hovercraft's speed, the PID will over-compensate since its increased torque, barely changes the hovercraft's speed (the wheels are spinning). This can easily be solved by measuring the wheel's speed. If the wheels go too fast the PID will slow down. But the problem is still there. Because if, for instance, the set speed is 10 m/s, while the hovercraft is bellow that set speed, let's say 5 m/s, the wheels will keep entering and exiting a spinning cycle until the hovercraft gets to the 10 m/s and the PID can slow it down definitely. Introducing a traction control could solve the problem but it's another complexity layer and the goal is to keep it simple and closer to manual as possible. Another solution is to fine tune the PID variables. Still, this solution doesn't achieve the desired goal because the PID always takes around the same time to achieve stability. This is bad because if the hovercraft is at full stop (0 m/s) and the user wants to go to 1 m/s, it will take as long as if he wanted to go to 10 m/s (granted that the PID is tuned not to make the wheels spin). This happens due to the proportional element of the PID. This element reacts proportionally to its stimulus. So if the set speed is 10 times faster it will

react with a torque 10 times stronger. This leads to a choice between making the wheel drift at high set speeds or take too long to get there when the set speed is low. A third choice is to be avoided. It relies on the user to set the speed gradually so it won't drift. This takes a system responsibility and puts it upon the user.

An easy, and rather elegant solution to this problem is to know at what torque the wheel starts drifting (the floor conditions aren't supposed to change so this torque is always the same) and apply a slightly smaller torque (so it won't spin) and keep that torque until the speed is close enough to the desired speed to the PID to take control and stabilize the torque/speed. This leads to the hovercraft getting to the desired speed as fast as possible because the optimal torque is always applied. One thing that wasn't taken into account was the fact that some cargo may have a maximum acceleration it can endure (fragile or liquid materials). But no information was given regarding it so it was decided that, at this point, the torque to apply, is always the optimum for achieving the desired speed.

5.5 – The Direction Controller

The direction of the wheels is also achieved by a motor that expects torque as an input. So, a solution similar to the speed controller was found to solve this problem. The user sets the positions and the PID controls the motor on behalf of the user so it gets there as soon as possible. But a new problem is found in here. The user is controlling direction (equivalent to position) and the PID is supposed to control torque (equivalent to acceleration). But a PID is only able to control one order of control above.

For instance, if the user sets the direction of the wheel at 90º and the wheel is at 45º, the PID will keep adding torque to the wheel until it gets to 90º. At this point it will start applying torque in the opposite direction. But the wheel now has speed, and with it, momentum. This momentum is enough to send the wheel up to a mirror state (90º + the starting 45º) where it will stop moving and then will start coming back again to the 90º. This will cause a pendular movement which will go on indefinitely.

So a stack of PIDs should be used on this case. A simplification was made and a P (proportional) was imbedded on the direction PID to accomplish the direction control.

5.6 – User tests

A user test composed of 11 users was conducted to find out what control layout was the best. This test can also provide possible issues with the existing schemes and, possibly, improvements to them.

5.6.1 – Concept

From all the sketched control layouts, only two were tested. The users drove the hovercraft using the two control layouts and their performance was recorded. Also comments and other qualitative data were documented during the tests to capture as many data as possible.

5.6.2 – Hypothesis

The test is intended to reveal which control layout is the best as well as to identify issues with both of them. The following is and hypothesis:

- The control layout 1 is better than the control layout 2.

"Better" needs to be defined, so the following hypothesis will replace the one above:

- The users take less time to complete a map with control layout 1 than with control layout 2.
- The users crash less with control layout 1 than with control layout 2.
- The users spend less time compensating for losses of control with control layout 1 than with control layout 2.

5.6.3 – Method

Two control layouts were tested and compared. Quantitative data was automatically recorded and more general and qualitative data was also recorded. Comments from the users were registered to help improve the controls.

The following data was recorded, every 16.(6) milliseconds:

- Time Stamp (in milliseconds since the current map start);
- The Hovercraft's position on the ground (two coordinates, in meters);
- The Hovercraft's orientation (a simple angle, in radians);
- The Front Wheel orientation (a simple angle, in degrees);
- The Back Wheel orientation (a simple angle, in degrees);
- The Hovercraft's linear velocity (in m/s);
- The Front Wheel angular velocity (in rad/sec);
- The Back Wheel angular velocity (in rad/sec);
- The Front Speed Control (in percentage);
- The Back Speed Control (in percentage);
- The Front Direction Control (in degrees);
- The Back Direction Control (in degrees).

This data is enough to replay what the user and the hovercraft was doing at any given moment of the test. It has the time stamp coupled with all special aspects of the hovercraft and all the positions of the user controls. From this data, a lot more can be extracted.

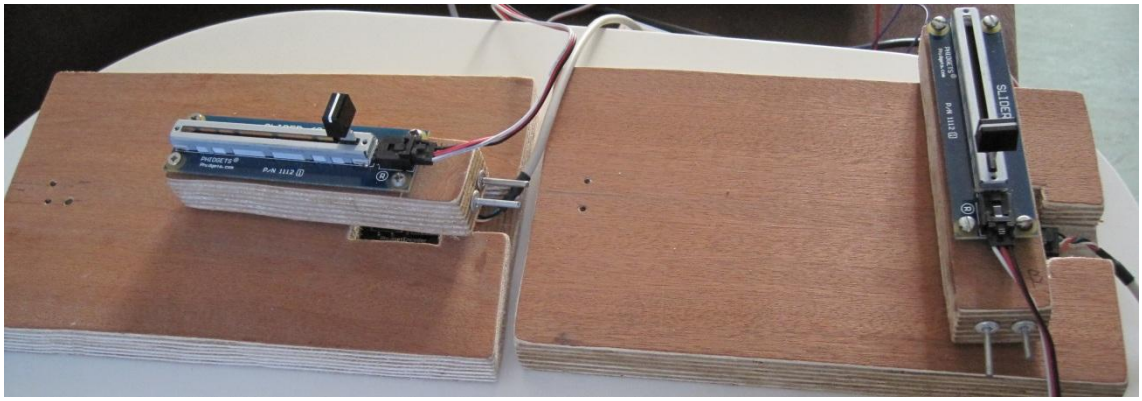
5.6.4 – Material

A normal laptop computer was used to run the simulation. Coupled with it were the controls which could be displaced in one of two layouts. The user could still move the controls around on the table, since the front and back controls were left separated. The maps used are considered material in a sense that they are important artifacts that are worth mention on the test.

5.6.4.1 – The Controls

Only two of the studied controls were built and tested. The two chosen controls are thought to be the best of the studied set to this problem. This does not mean that a third design would not be better or even a design that was not considered. The number of chosen controls is two because of how hard it is to build the hardware prototypes and how long it takes to do the user studies. With more available time and resources more designs would have been build and tested. Also several iterations of the best designs would have been built.

The hardware was built in such a way to accommodate for easy switching between the two layouts. This may have led to less stability of the prototypes but was necessary because of the number of times the layouts have to be switched. This problem could have been eliminated if two sets of prototypes were built but this, effectively, doubles the financial cost of the prototype. On this case it was decided to go with reassembleable controls.

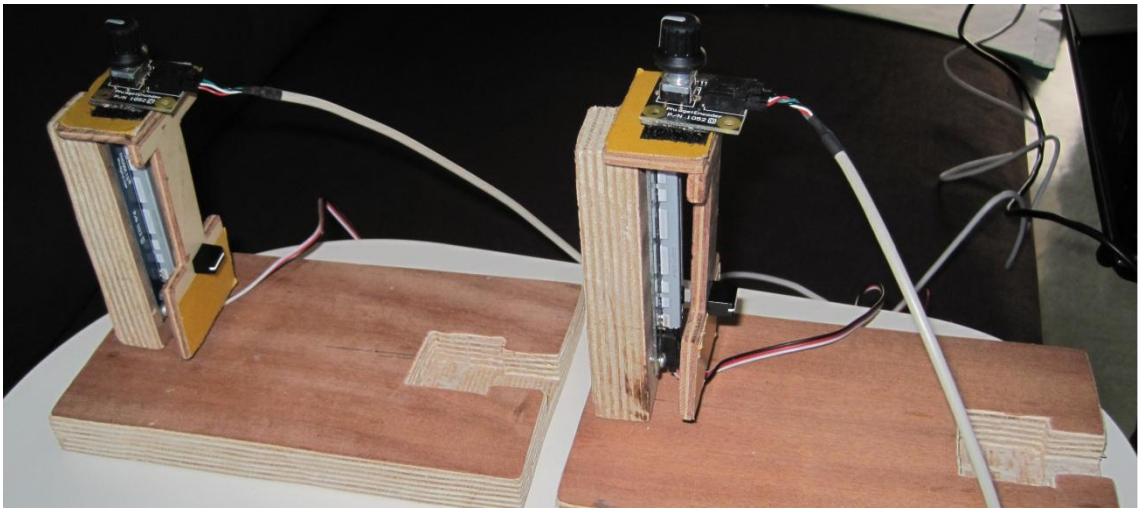


Img. 9- Control Layout 1

The first layout hides the rotary control and gives direct handling of the speed control to the user. The user controls the speed of the wheel by moving the handle away from the center and the orientation by rotating the whole control around the center point. This longer arm gives the user better control over the direction. It also directly maps the movement with the control since the user will move the control to where he wants the hovercraft to move and if he wants it to go faster he should move the control farther.

This prototype suffered from four building issues:

- The rotation point is a little bit fragile and uncareful fast movements may damage the piece of wood that holds the speed control.
- The rotary control has a button. That button works by pressing the rotary axis itself. It is used on the simulation to set the orientation to 0º, allowing for easy reset. It has proven to be an issue since a slight pressure by the hand might push that button resetting the orientation and making the hovercraft go out of control.
- The handle is not round which would allow for better rotation. Also, a lose handle would further increase the rotation ability but, possibly, reduce precision.
- Since the whole speed control rotates, its wires rotate with it, leading to stretched and crossing wires.



Img. 10 - Control Layout 2

The second layout, as opposing to the first layout, hides the speed control and gives the user direct handling of the direction control. The user controls the speed by moving the control up and down (up means faster but could be switched if the tests point that way). To turn the wheels, the user turns the handle of the rotary control.

This prototype suffered from X issues:

- The handle was too small, giving bad grasping to the user and demanding very fine movements.
- Since the speed controls are set vertically, the user will end up resting his hands on it lowering the speed. This issue can be mitigated by reducing the length of the speed controller course and / or introducing a rest position for the hands.
- The rotary control was hard to fix in a way that would allow for the required reassemble. Velcro was the solution used and proved enough but users were noted to be worrying with it, which should have not happen.

Since the rotary control does not move and the speed control only moves slight vertically, the wires are not an issue on this design.

5.6.4.2 – The maps

A set of maps were designed to allow the testing of the controls. There are four maps without obstacles, four with obstacles and a final test. The final test allows the user to test all that was learned on the previous maps on a tight, more complex map. The maps are described on the following paragraphs.

Map 1 – This map's objective is placed right ahead of the user. This allows the user to learn how the speed controls work without having to worry about the direction of the hovercraft.

Map 2 – This map introduces direction but on a simple way. The objective is slightly out of the way requiring the user to start using the direction. Since it is a simple change of directions, the user may use only one control.

Map 3 – The objective is now place on the side of the hovercraft. The user may just use one direction control to get to the objective but the map incentives the user to try the second controller. If he does so and has learned enough of how the hovercraft works, he may try to point each wheel to opposing directions, getting a faster 90º rotation. Or he may try a strafe move, pointing both wheels to the objective and moving the hovercraft sideways to the objective.

Map 4 – On this map, the objective is set behind the hovercraft while the hovercraft is set very close the wall and facing it. This is designed to make the user think of other ways to drive the hovercraft so he can get away from this situation without colliding with the wall. The user may perform one 180º (the same way as the above described 90º but rotating more) for the first time or move back (pointing both wheels back), also for the first time.

Map 5 – This is the first of a series of maps with obstacles. This map is the second map with an obstacle that requires a sharper change of direction of 90º, around a corner, teaching cornering with the hovercraft.

Map 6 – This is also a remake of an old map adding an obstacle. It is the first map with an obstacle between the hovercraft and the objective that the user will have to go around to get to. It requires a long, controlled curve.

Map 7 – This map, basically, increases the difficulty of the 90º cornering making it 180º. If the user has learned enough, he shall feel the need to use the back wheel to increase the cornering performance.

Map 8 – This is the final training map. It requires the user to make a zigzag. It is composed of a 90º curve followed by a 180º counter-curve. It is also slightly tighter in preparation of the final test. The user that uses the back wheel may start to feel its advantages here.

Final Map – The final map is a tight and more complex map that puts all the learning to the test. It starts on a position that requires a 90º turn or a strafe move. The best way to start is a strafe move since it will avoid a tight 180º turning. It follows with a zigzag, a 180º tight curve and another zigzag. In all of those the user greatly benefits from the correct use of the back wheel. If the user does not use it, collisions are a very high probability. The map ends with a simple choice of which side to go through to the objective. One side is safer and easier while the other requires a thigh 90º turn. But that 90º turn is the shorter path. This is the only time where the user has to make a choice coinciding with when the user is supposed to be at the pinnacle of his learning (during the test). It is a simple test to see if the user is using too much brainpower to drive the hovercraft.

Given the increasingly difficulty of each map, it was decided to serve the maps on order instead on randomly to avoid the user being given a too hard map without proper training or a too easy map after he has gone through the whole test-set.

5.6.5 – Procedures

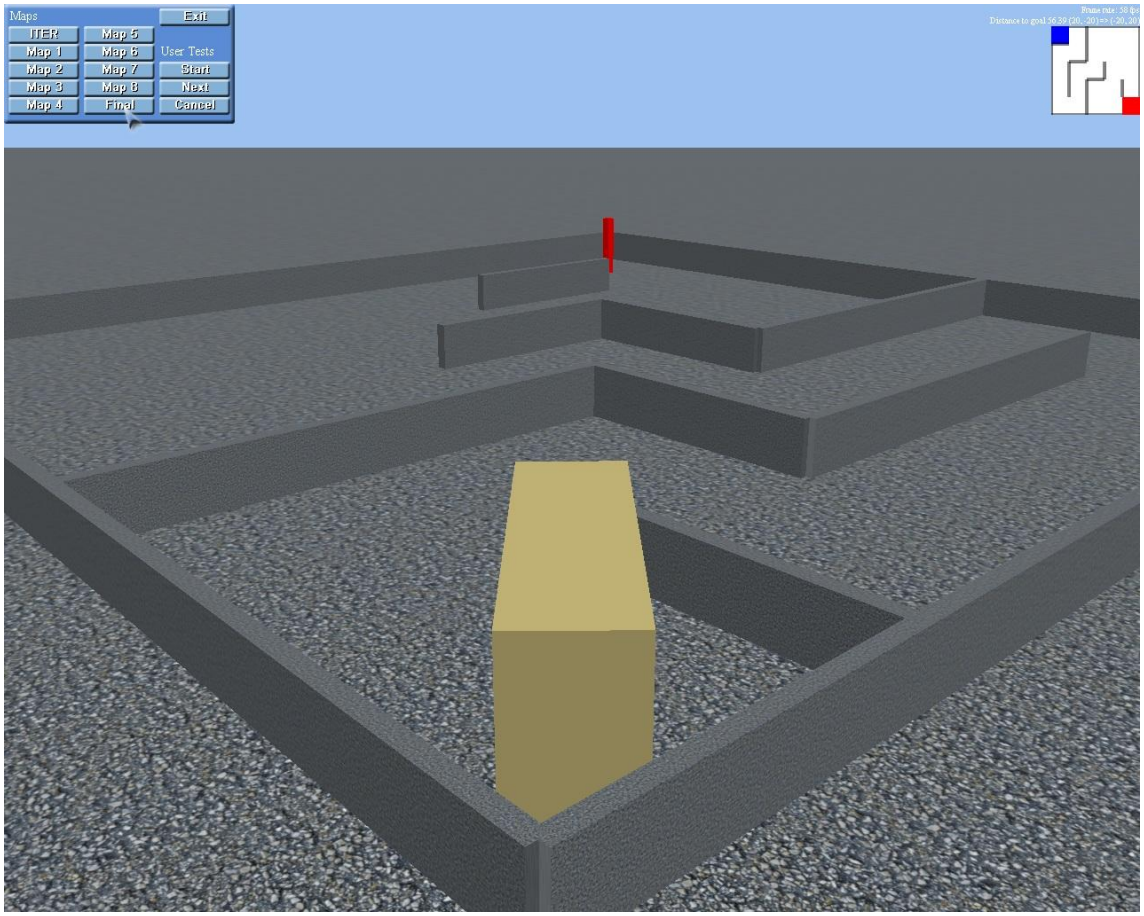
Material Preparation – The hardware is assembled and all relevant software is initialized. A quick check is made on the hardware operability, correct position (front and back) and the orientation's accuracy.

User introduction – The user is introduced to several concepts without being given more information than needed. The following information is given to the user, in an informal way, but respecting the order:

1. This is a simulation where the user will drive a 10 (metric) Ton hovercraft with a length of approximately 8m long.
2. The hovercraft has two wheels located one in the front another on the back.
3. Each wheel has 360 º rotation capabilities.
4. The user will have control over the speed and the direction of each wheel.
5. The user is presented to the controllers and each controller is identified as well as is function, in a general, intentionally vague way.
    a. The user is informed that the speed controller sets the speed directly and the user does not need to worry about how the hovercraft achieves it (this is said so the user does not try apply the concept of driving a car to the driving of the hovercraft; this concept is dangerously present on most of the users testers which lie around 20 year old).
    b. The user is also informed that the rotation controls the direction of the wheels. No further information is said about the direction control to allow the study of how the users understand and learn how it works.
6. The user is now informed that he will have to drive the hovercraft through a series of "maps". He is informed that his objective is to get to

the red box. He is then quickly presented with a simple hand-sketch of the "maps". The maps are tiny and by hand in hope of introducing only the concept of "maps" and objective and not allowing for learning of the "maps". The user is also informed that he will have three seconds on the start of each "map" to see what he has to do.

User first contact – The user is given a moment to try out the controls on a clean, objectiveless "map". When the user feels ready, he should request the starting of the tests.



Img. 11 - A Screenshot of the simulation runing on the final map

Tests – The user goes through the 9 above mentioned "maps" in order while the computer logs the relevant data and observations are documented on paper. No aid is given at this point.

Controls switch – Once the user finishes all 9 maps the controls layout are switched and the tests continue from the step *User first contact* on.

Test finalization – The user is asked which layout he thinks is better and why. He is also given the opportunity of giving additional comments and thoughts about the test. At this point user is thanked and a new test starts (if there are any more users).

The users alternate the layout they start with. This not only tries to compensate for the learning of the layouts but also avoids switching the layout between users, giving better timing.

No interface is used to give aid or training on the controls. The only feedback that the user receives is the physical feedback from the hovercraft itself. This was a decision made to allow an understanding of how fast and well the users understand the controls layout and mapping to the hovercraft. This also allows the identification of possible issues on the design of the controls.

5.6.6 – Results

From all the data, only three aspects were chosen to compare both control layouts and this data is only taken from the final map of each layout. This reduces different results between users that learn faster from those that learn slower. And this must be done because the users do not only need to learn the control layout. They also need to learn the physical concept of a two 360º wheeled hovercraft, which is not a common one. And this concept is not what is being tested here.

The (effective) time to complete is defined as the time it takes to the user to complete the current map, starting as soon as the hovercraft starts moving (some users like to take longer to look at the map while not moving the hovercraft).

The number of crashes is defined as the number of times the hovercraft acceleration was above 0.5 m/s^2. The PhysX could have logged collisions but it was only while the tests were being conducted that the need to log that appeared. To keep the data coherent no change was made during the tests and

a way to detect crashes was developed. Any acceleration above the normal hovercraft's acceleration is considered a collision. This leaves small collisions and wall scratches out. It is worth saying that not all the users cared about collisions. Some even used collisions as an aid to turn the hovercraft. This may reduce or even invalidate the importance of this parameter.

The time spent compensating is defined as the percentage of the time that the users spent compensating for over-steering (a sign of loss of control). This is found by searching for entries were the hovercraft is turning in one direction and the user is turning the controls to compensate that movement (the front wheel must be turned on the opposite direction of the hovercraft's rotation while the back wheel must be turned on the same direction).

The amount of time spent using the controls can be seen as an indicator of back driving. If the user keeps moving the controls more than he should, that means that the hovercraft is not under control. But would require a good definition of what is enough to control the hovercraft, what is an excess of usage and what is a lack of usage.

5.6.6.1 – Data

The effective time it takes to complete the final map is, most of the times, longer on the layout two (seven out of ten cases). The average of these times tells us that the layout two is slightly more time consuming than layout one.

After calculating the delta between the two layouts (L2 – L1), the average of that delta is above zero (this means that layout two takes longer) and the confidence interval, at 95%, is fully above zero.

Despite the fact that everything points to Layout 2 being more time consuming than Layout 1, the T-Tests indicate a large and unequal variance. This variance sets P, for a paired T-Test, at 0.0811. Making another T-Test, this time taking into account that the samples have unequal variances, pulls P down to 0.0486. This means that these results can be regarded as not as pure chance, but barely.

The number of collisions is higher on Layout 2. But it is not very far above. There were a total of 55 accounted collisions on the final map for Layout 2 against a total of 48 collisions under the same circumstances for Layout 1. Note that smaller collisions are not taken into account and could make a difference in either direction.

After the delta between the number of collisions and the confidence interval, at 95%, was calculated, the mean barely stays above zero (this means that Layout 2 had more collisions that Layout 1) and the confidence interval is well in both sides. This means that we cannot be sure that the Layout 2 is indeed more prone to crashes than Layout 1.

After conducting the T-Tests, a P of 0.6398 is achieved. This is pretty poor result which takes out the credibility of this test. Not even the T-Test that takes unequal variances helped here. An even worse P of 0.6748 is achieved.

The time that users spent compensating points, once again, for a better performance of Layout 1. Users spend an average 2.82% of the time compensating for hovercraft's overturn using Layout 2 while they spend 2.37% of the time doing the same with Layout 1. Again the difference is slight.

The delta (L2 – L1) was also calculated here as well as the confidence interval. The mean stays above zero (this means that Layout 2 is more prone to loss of control than Layout 1) and the interval stays above zero, with one edge touching the zero. A conclusion can be made that Layout 2 is, probably, worse than Layout 1 on this aspect but, there is a slight chance it may not be this way.

The T-Tests are better than the previous ones but still discard the credibility of this test. P, for a paired T-Test is at 0.3266. And again, if we take into account the unequal variance, it grows up to 0.4575.

5.6.6.2 – Discussion

All the three tests indicate that Layout 1 is slightly better than layout 2. But the T-Tests and the confidence interval indicate that this may have been just an accident. Despite the high probability of these results being an accident and one confidence interval largely overlapping both sides of the graph, three tests out of three pointing that Layout 1 is better than Layout 2 must have some significance. Together with this tests one should take into account most users verbally showed their preference towards Layout 1, while showing some fatigue and/or discomfort towards Layout 2.

This leads to a conclusion that Layout 1 is, probably, slightly better than Layout 2. A bigger population sample could have helped to achieve more credible results as only ten users tested the two layouts. This gives five untrained users to each.

If both Layouts were improved and, possibly, redesigned these results may change and even invert. So, more though and develop time should be given to the controls layout design before a test could be done that could definitely find the best control layout.

Still, even if this test did not find a definite candidate, issues on the controllers were found and the room for improvement is there. So, this test should not be regarded as a decision between two layouts but as a point of start for a better design.

5.6.7 – Improvements

The hardware bought – First of all, some issues were inherited from the hardware bought.

The speed control has a long course which was not desired for the second layout. Since this layout relies on vertical movement, fatigue was rapidly noted since the hand could not rest. A smaller course would have enabled the hand to stay close to the base enabling it to rest.

The rotary control has a small handle which requires the user to have finer movements. A bigger handle would have helped, increasing the linear distance needed per angle. The rotary control has a jump of 4.5º which was thought to be a problem but ended up not being noticed. Since it uses continuous/relative rotation, its pooling frequency may jump steps if one rotates it too fast. So the user should not rotate too quickly. This is the reason

why the button present on this control was used to reset the orientation to 0º. This avoids having to restart the simulation and, consequently, the tests.

The prototype built – The prototype is a functional one but was built using relative low-tech material. Besides the electronics used, the bases where those electronics stand are made of hand shaped wood and even velcro.

The decision to build a prototype that could easily be switched between the two layouts might have ended on having a weaker prototype. This might have affected results since it may have penalized more one layout than the other. Also, the prototype itself has weak points (such as the whole where the rotary control plugs in) that may introduce a mechanical lag and jerkiness.

The wires also introduced some trouble as they could decrease the liberty of the movements and its tension could pull the controls to a position the user did not set.

The bulkiness of the whole control system could also decrease user performance. But this alone is not troublesome. Very small handles to such big controls might have been the problem. Some users complained that the handles, specially on Layout 2 were too small to give them enough precision over the controls. Respectively, the Layout 1 had square handles which did not handle rotation movement very well. An even better handle would be one that could rotate around itself. This would allow for a smother movement as no friction would appear between the user and the controller.

The population – The user-tests were run using 11 users. One of each did not complete the tests so he is not taken into account. This leaves 10 users for two layouts. This makes up for a quite small population. A bigger population sample would have helped achieve more significant results. The number might also compensate for the fact that the users more used to this kind of technology (joysticks / games) may have been more on one side than another.

It should be taken into account that a random population was used to conduct the tests while, on the real set, a trained population with a good background in driving, joysticks, technology and physics would be used. But a random population is helpful to help identify if the controls are indeed intuitive since a trained population may overcome any difficulty using their training. Still, the number of elements may have not been enough to absorb the different skill-set present on the population sample giving the results it did.

Conclusion

The simulation ended up having enough physical accuracy to give the users a taste of how to drive the hovercraft using the controls built. But the visualization was not enough to give the users neither the sense of size nor the atmosphere of driving the real thing on the real place. There is room for improvement on this part of the simulation but it is already good enough to test control layouts and the physics of driving such a vehicle with satisfaction. The fact that the hardware program is separated from the simulation also helps on future layouts tests with different hardware than the ones used during this thesis.

The first Layout won the tests as the best layout but just barely. The difference is not big enough and the quality of the tests, proven by the T-Tests, is not good enough to take these results as definitive. Despite this, several issues were identified which may help on future development of others controls layouts. Even maintaining the current layouts, better prototypes and a broader population sample may produce better results, helping identify the best candidate and even more issues.

Bibliography

1. **Lewis, F.L.** *Applied Optimal Control and Estimation.* s.l. : Prentice-Hall, 1992.

2. Tutorial 15: Control Theory, Part I. [Online] http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4534376&isnumber=4534362.

3. **Rojas, Raul.** *PID.* [Online] http://robocup.mi.fu-berlin.de/buch/pid.pdf.

4. A Brief History on Feedback Control. [Online] http://www.theorem.net/theorem/lewis1.html.

5. Application of Classical Controller Design. [Online] http://academic.csuohio.edu/cact/gao/eec640/Sample_report.pdf.

6. Loop Tunning. [Online] http://www.jashaw.com/pid/tutorial/pid6.html.

7. *Expert Tunner.* [Online] http://www.expertune.com/standard.asp.

8. **Wickens, Christopher D., et al., et al.** *Human Factors Engineering.* s.l. : Pearson, 2004. 0-13-122917-6.

9. The Free Online Dictionary. [Online] www.thefreedictionary.com.

10. **P. Robuffo Giordano, H. Deusch, J. Lachele, and H. H. Bulthoff.** *Visual-Vestibular Feedback for Enhanced Situational Awareness in Teleoperation of UAVs.*

11. **Brownbridge, Jason.** *Teleoperation of Rescue Robots in Urban Search and Rescue Tasks.* s.l. : University of Cape Town.

12. **Michael Baker, Robert Casey, Brenden Keyes, and Holly A. Yanco.** *Improved Interfaces for Human-Robot Interaction in Urban Search and Rescue.* s.l. : University of Massachusetts Lowell.

13. *Global Positioning System.* [Online] http://www.gps.gov/.

14. **Minch, Dr. Robert P.** *Privacy Issues in Location-Aware Mobile Devices.* Boise State University : s.n.

15. GSM location. [Online] http://www.gsm-modem.de/gsm-location.html.

16. *GPS Explained.* [Online] http://www.kowoma.de/en/gps/accuracy.htm.

17. *Encyclopedia of Laser Physics and Technology.* [Online] http://www.rp-photonics.com/distance_measurements_with_lasers.html.

18. *The History of Sonar.* [Online] http://inventors.about.com/od/sstartinventions/a/sonar_history.htm.

19. *Radar.* [Online] http://www.radarworld.org/.

20. *Inercial Sensor.* [Online] http://www.analog.com/en/sensors/inertial-sensors/adis16405/products/product.html.

21. *Gyroscopes.* [Online] http://www.gyroscopes.org.

22. *Accelerometers.* [Online] http://www.omega.com/prodinfo/accelerometers.html.

23. *Magnetometers.* [Online] http://www.gemsys.ca/Magnetometers/16_about_magnetometers.htm.

24. **Salvendy, Gavriel.** *Human Factors and Ergonomics.* s.l. : Salvendy, 2006.

25. *LEDs.* [Online] http://www.kpsec.freeuk.com/components/led.htm.

26. *What is Stereo Vision.* [Online] http://www.vision3d.com/stereo.html.

27. **Rolland, Jannick P. and Fuchs, Henry.** *Optical Versus Video See-Through Head-Mounted Displays in Medical Visualization.* University of Central Florida; University of North Carolina : s.n.

28. **John HU, Chu-Yin CHANG, Neil TARDELLA, James ENGLISH, Janey PRATT.** *Effectiveness of Haptic Feedback in Open Surgery Simulation and Training Systems.* s.l. : Energid Technologies Corporation; Massachusetts General Hospital.

29. **Burdea, Grigore C.** *Haptic Feedback for Virtual Reality.* s.l. : The State University of New Jersey.

30. **Yoo, Martin Faust and Yong-Ho.** *Haptic Feedback in Pervasive Games.* University of Bremen : s.n.

31. Tesla Touch. [Online] http://www.teslatouch.com/.

32. *What is Force-Feedback.* [Online] http://krafttelerobotics.com/force_feedback.html.

33. *Haptics: Texture.* [Online] http://www.virtualworldlets.net/Resources/Hosted/Resource.php?Name=Haptics-Texture.

34. [book auth.] Jenny Preece et al. *Human-Computer Interaction.* s.l. : Pearson Education Limited, 1994, Input.

35. *Discrete and Continuous Probability Distributions.* [Online] http://stattrek.com/Lesson2/DiscreteContinuous.aspx?Tutorial=Stat.

36. *Digital Vs Analog.* [Online] http://www.skullbox.net/dva.php.

37. 3D Connexion. [Online] http://www.3dconnexion.com.

38. 3D Motion For You. [Online] http://www.3dmotion4u.com/products.php.

39. Falcon. [Online] http://home.novint.com/products/novint_falcon.php.

40. *History of Human-Computer interaction.* [Online] http://www.cs.cmu.edu/~amulet/papers/uihistory.tr.html.

41. *TouchScreen Archive.* [Online] http://solutions.3m.com/wps/portal/3M/en_US/TouchSystems/TouchScreen/Information/Media/PressReleases/Archive/?PC_7_RJH9U52300FA602N9RSR991OI3_assetId=1180610310216.

42. *Kinect.* [Online] http://www.xbox.com/en-US/kinect.

43. *Touch Usability.* [Online] http://www.touchusability.com/blog/2010/9/12/how-not-to-do-gestures-in-your-iphone-app.html.

44. *Voice Recognition.* [Online] http://www.landmark.edu/institute/assistive_technology/voice_recognition.html.

45. Latency. [Online] http://en.wikipedia.org/wiki/Latency_(engineering).

46. **Rony Kay, Ph.D.** *Pragmatic Network Latency Engineering - Fundamental Facts and Analysis.* cPacket Networks : s.n.

47. How far is Mars from the Earth. [Online] http://www.universetoday.com/guide-to-space/mars/how-far-is-mars-from-the-earth.

48. Bandwidth. [Online] www.malwarecity.com/site/Main/listDictionary/B/.

49. *ITER.* [Online] http://www.iter.org.

50. IST - Remote Handling on ITER. [Online] http://www.ipfn.ist.utl.pt/rh.

51. ITER Countries. [Online] http://draft.isr.ist.utl.pt/cla/iter_paris_acordo.shtml.

52. ITER cost. [Online] http://www.bbc.co.uk/news/science-environment-10793883.

53. Fusion for Europe. [Online] http://fusionforenergy.europa.eu/.

54. *OpenGL.* [Online] http://www.opengl.org.

55. *DirectX.* [Online] http://msdn.microsoft.com/en-us/directx.

56. *SDL.* [Online] http://www.libsdl.org.

57. *OGRE.* [Online] http://www.ogre3d.org.

58. *NVidia.* [Online] http://www.nvidia.com.

59. *AMD.* [Online] http://www.amd.com/uk/Pages/AMDHomePage.aspx.

60. PhysX GPUs. [Online] http://www.nvidia.com/object/physx_gpus.html.

61. PhysX DCC Plug-Ins. [Online] http://developer.nvidia.com/object/physx_dcc_plugins.html.

62. 2012 uses Bullet. [Online] http://bulletphysics.org/wordpress/?p=132.

63. *PhysX.* [Online] http://www.nvidia.com/object/physx_new.html.

64. *Havok.* [Online] http://www.havok.com.

65. *Bullet.* [Online] http://bulletphysics.org.

66. *Newton.* [Online] http://newtondynamics.com.

67. *ODE.* [Online] http://www.ode.org.

68. Physics Engine Comparison. [Online] http://physxinfo.com/articles/?page_id=154.

69. GPGPU History - ATI. [Online] http://www.amd.com/US/PRODUCTS/TECHNOLOGIES/STREAM-TECHNOLOGY/OPENCL/Pages/gpgpu-history.aspx.

70. *liblo.* [Online] http://liblo.sourceforge.net/.

71. *OSC.* [Online] http://opensoundcontrol.org/.

72. *Arduino.* [Online] http://www.arduino.cc/.

73. Title Image. [Online] http://agaver.files.wordpress.com/2009/09/matrix_oracles_apt_psychokinesis_no_spoon1.jpg.

74. **Okamura, A.M.** *Methods for haptic feedback in teleoperated robot-assisted surgery.* Department of Mechanical Engineering, The Johns Hopkins University. Baltimore, Maryland, USA : s.n. http://www.emeraldinsight.com/Insight/ViewContentServlet?Filename=Published/EmeraldFullTextArticle/Articles/0490310606.html.

75. Noise. [Online] http://en.wikipedia.org/wiki/Noise.

76. **Mesures, Bureau International des Poids et.** The International System of Units (SI). [Online] 8th edition. http://www.bipm.org/utils/common/pdf/si_brochure_8_en.pdf}.

77. Windows API. [Online] http://msdn.microsoft.com/en-us/windows/ff404219.aspx.

78. *Phydgets.* [Online] http://www.phidgets.com/.

79. Watt's Steam Engine Governor. [Online] http://blog.cpas.anu.edu.au/diffusion/files/2010/02/Watt-governor.jpg.

80. *NASA - South Polar First Look.* [Online] http://www.nasa.gov/mission_pages/Mini-RF/news/2009-01-16_south_polar.html.

Acknowledgments

First and mostly of all I want to thank my supervisor Ian Oakley for guiding and helping me through this long and arduous ride. It has been a very interesting and productive year of my life and I learnt a lot from you. I know you have a lot of students under your supervision but even so you manage to have more than enough time to each and every one of them. Keep up the good work.

I want to give a special thanks to all my test users that helped me complete my study. They also actively helped me to better understand my mistakes and find ways to correct them by giving hints and their thoughts on my work.

- André Ferreira
- Cláudia Nunes
- Clinton Jorge
- Elsa Ferreira
- Joana França
- Roberto Mendonça
- Ruben Mendonça
- Sérgio Freitas
- Soraia Ferreira
- Tiago Gonçalves
- Tiago Sousa

(Ordered alphabetically)

Not directly related to this thesis but very important non-the-less I want to thank my parents who worked very hard all their lives so I could get to where I am today. I don't think I would be able to get this far without them.

I also want to thank my girlfriend, Elsa Ferreira, for always being there for me. I'm sorry for when I had to choose this thesis over being with you.

Finally, I want to thank every single person not listed here that helped me, directly or indirectly through my life up to this big achievement.

Appendices

Appendix A – Raw Test Logging

This is the first 5 time-steps of a logging file from a real test user. Note that this file is not supposed to be directly read by humans. Despite this, it is in such a format (csv) that allows easily importation into a spreadsheet.

The first line identifies what will be logged. The following lines contain the values identified on the first line, on the time step the simulation is at the moment. Since the time-step was is set to 60 Hz, this means a new entry every 16.(6) milliseconds.

```
TimeStamp,HovecraftPosition.x,HovercraftPosition.z,Hovercra
ftOrientation,FrontWheelOrientation,BackWheelOrientation,Ho
vercraftLinearVelocity,FrontWheelAngularVelocity,BackWheelA
ngularVelocity,FrontSpeed,BackSpeed,FrontDirection,BackDire
ction
0,0.000000,-
20.000000,0.000000,0.000000,0.000000,0.000000,0.000000,0.00
0000,0.000000,9.000000,0.000000,0.000000
1,-0.000000,-
20.000000,0.000000,0.039565,0.000000,0.000000,0.000000,0.22
3232,0.000000,9.000000,0.000000,0.000000
17,-0.000000,-
20.000000,0.000000,0.104678,0.000000,0.000000,0.000000,-
0.879044,0.000000,9.000000,0.000000,0.000000
33,0.000000,-
20.000002,0.000000,0.201741,0.000000,0.000021,0.000712,0.27
9631,0.000000,9.000000,0.000000,0.000000
```

(This file continues. To see a full file, please look into the CD that accompanies this thesis.)

Appendix B – Digested Test Logging

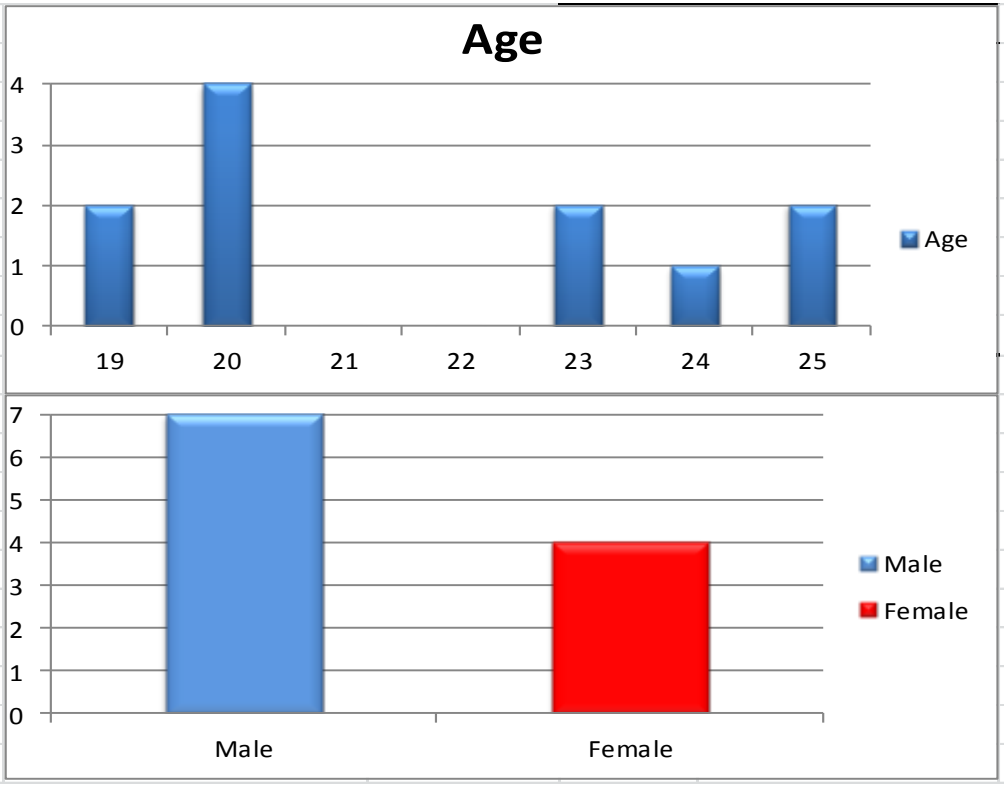
Following is an example of part of a human-readable extract from a log file used to log the user-tests. This is a resume of the bigger, more detailed log file that the simulation program generates every time-step (please see the previous page). Together with this file, another file is generated. That second file is similar to the one showed on the previous page but containing this new, threated information. The objective of that file is to allow easily importation into a spread-sheet while this one allows for quick human check.

```
\Logs\TestUser\4883325 - Map1.csv

  Number of entries: 2335
  Time to Complete: 38736msec | 38sec
  Time to Complete: 36146msec | 36sec (Effective)
  Mean Speed: 0.934309 m/s
  Number of crashes: 0
  Front direction command usage: 2.62
  Back direction command usage: 0.17
  Front speed command usage: 1.33
  Back speed command usage: 4.29
  Time spent compensating: 0.90
  Front compensation: 0.90
  Back compensation: 0.00

\Logs\TestUser\4931440 - Map2.csv
  Number of entries: 3185
  Time to Complete: 52847msec | 52sec
  Time to Complete: 52847msec | 52sec (Effective)
  Mean Speed: 0.973672 m/s
  Number of crashes: 0
  Front direction command usage: 0.91
  Back direction command usage: 0.53
  Front speed command usage: 1.35
  Back speed command usage: 1.60
  Time spent compensating: 0.57
  Front compensation: 0.35
  Back compensation: 0.22
```

(This file continues. To see a full file, please look into the CD that accompanies this thesis.)

Appendix C – Users (testers) Demographics

Age	Sex
23	M
23	M
25	M
19	F
20	F
19	F
20	M
20	F
24	M
25	M
20	M
Mean	Male
21.63636	7
Range	Female
19-25	4



Age	
Mean	21.63636364
Standard Error	0.716972515
Median	20
Mode	20
Standard Deviation	2.377928816
Sample Variation	5.654545455
Kurtosis	-1.761664995
Skewness	0.36554459
Range	6
Minimum	19
Maximum	25
Sum	238
Count	11
Confidence Level (95%)	1.597514316

Appendix D – Compiled User-Test Digested Results (1 user per page)

Units			unit	msec	msec	m/s	unit	m/s^2	m/s^2	m/s^2	%	%	%	%	%	%	%
User	Commands Layout	Map	Entries	Time to Complete	Time to Complete (Effective)	Average Speed	Number of Crashes	Most Violent Crash	Average Crash Acceleration	Crash Acceleration Sum	Front Direction Command Usage	Back Direction Command Usage	Front Speed Command Usage	Back Speed Command Usage	Time Spent Compensating	Front Compensation	Back Compensation
1	1	1	2,093	34,719	34,303	1.028829	0	0	0	0	0.239234	0	1.626794	1.722488	0.143541	0.143541	0
		2	3,156	52,364	52,364	1.034845	0	0	0	0	1.014906	1.395496	0.190295	0.126863	0.444022	0.285442	0.158579
		3	2,517	41,759	41,759	0.950848	0	0	0	0	2.505967	1.392204	0.079554	0.079554	1.113763	0.875099	0.238663
		4	2,821	46,806	46,806	0.835206	2	8.379375	5.796969	11.593938	3.655075	3.016324	0	0.141945	1.348474	1.064585	0.283889
		5	3,358	55,726	55,726	1.065505	0	0	0	0	0.804769	1.341282	0.178838	0.119225	0.923994	0.536513	0.387481
		6	3,241	53,779	53,779	0.993422	0	0	0	0	2.594194	5.095738	0	0.061767	2.254478	0.988264	1.266214
		7	4,558	75,639	75,639	1.048947	0	0	0	0	2.568606	1.097695	0.087816	0.043908	1.624588	1.29528	0.329308
		8	4,642	77,038	77,038	1.014114	0	0	0	0	2.328088	2.996335	0.043113	0.043113	1.530502	0.905368	0.625135
		Final	9,761	162,034	162,034	0.755673	7	13.174626	5.766285	40.363996	5.083009	6.579217	0.10248	0.081984	3.863497	2.275056	1.598688
	2	1	2,056	34,106	34,106	1.044309	0	0	0	0	0	0	0.097418	0	0	0	0
		2	3,306	54,857	54,857	0.972223	0	0	0	0	5.207387	4.026642	0.121102	0.060551	3.118377	2.179837	0.999092
		3	2,326	38,592	38,592	0.962333	0	0	0	0	2.152389	1.506672	0	0.086096	1.54972	1.033147	0.516573
		4	2,937	48,733	48,733	0.81275	1	3.128814	3.128814	3.128814	3.306067	1.976823	0.272665	0.068166	2.04499	1.499659	0.545331
		5	3,275	54,349	54,349	1.019803	0	0	0	0	4.187042	1.314181	0.061125	0.122249	2.322738	1.925428	0.397311
		6	3,543	58,797	58,797	0.886167	0	0	0	0	4.887006	3.022599	0	10.564972	2.655367	1.977401	0.677966
		7	4,532	75,215	75,215	1.009807	0	0	0	0	2.340473	2.958711	0.13248	18.63546	1.766394	1.015677	0.750718
		8	4,779	79,326	79,326	0.961686	1	1.020687	1.020687	1.020687	3.203518	3.140704	0.125628	20.01675	1.926298	1.360972	0.565327
		Final	8,273	137,335	137,335	0.823836	3	4.640312	2.298884	6.896652	4.860943	4.292624	3.482467	22.623942	2.394196	1.789601	0.616687

2	**2**	1	2,199	36,478	36,478	0.981582	0	0	0	0	2.823315	0.227687	22.26776	8.287796	1.092896	1.047359	0.045537
		2	3,082	51,139	51,139	1.028665	0	0	0	0	1.234167	0.129912	18.934719	7.892173	0.779474	0.68204	0.097434
		3	2,567	42,587	42,587	1.01972	0	0	0	0	2.301092	0.156006	19.149766	0.468019	0.936037	0.936037	0
		4	18,458	306,456	306,456	0.5646	12	30.850822	9.526235	114.31482	7.320509	1.804389	13.253861	2.638851	1.804389	1.452181	0.53644
		5	10,341	171,675	171,675	0.765738	1	4.360293	4.360293	4.360293	5.697427	0.764171	0.116077	4.817179	1.567034	1.218804	0.34823
		6	5,789	96,090	96,090	0.753638	4	9.150704	5.674829	22.699315	5.288628	0.414794	0.138265	0.846872	1.711027	1.641894	0.069132
		7	10,101	167,683	167,683	0.71905	4	3.459765	2.119971	8.479882	4.901961	1.81224	14.32957	8.011487	1.733016	1.37651	0.386215
		8	13,482	223,829	223,829	0.751798	10	27.61681	10.78655	107.865496	5.423251	1.446695	21.960086	14.897248	0.942206	0.882855	0.066771
		Final	12,421	206,202	206,202	0.792791	11	39.562374	17.762192	195.384109	2.544693	0	19.922693	10.992108	0.885811	0.885811	0
	1	1	2,169	35,977	35,977	1.046069	0	0	0	0	0.415512	0	0.092336	0	0.138504	0.138504	0
		2	7,103	117,914	117,914	0.982059	2	9.297823	5.551558	11.103117	1.197183	0.183099	0.225352	0.084507	0.56338	0.492958	0.070423
		3	2,743	45,511	45,511	1.058666	0	0	0	0	0.839416	0	0.218978	0.218978	0.255474	0.255474	0
		4	3,547	58,866	58,866	0.933402	1	4.567588	4.567588	4.567588	1.269752	0.08465	0.112867	0.1693	0.564334	0.507901	0.056433
		5	3,492	57,953	57,953	1.089149	0	0	0	0	0.515907	0.028662	0.229292	0.057323	0.143308	0.143308	0
		6	3,140	52,270	52,270	1.076232	0	0	0	0	0.892573	0	0.12751	0.063755	0.255021	0.255021	0
		7	4,665	77,432	77,432	1.104196	0	0	0	0	0.3003	0	0.1287	0.0429	0.10725	0.10725	0
		8	4,975	82,574	82,574	1.082931	0	0	0	0	0.804505	0.020113	0.120676	0.120676	0.221239	0.221239	0
		Final	7,694	127,718	127,718	0.959301	5	38.507759	17.857364	89.286822	2.015343	0	0.780133	0.104018	0.520088	0.520088	0

3	1	1	2,335	38,736	36,146	0.934309	0	0	0	0	2.61578	0.171527	1.329331	4.288165	0.900515	0.900515	0
		2	3,185	52,847	52,847	0.973672	0	0	0	0	0.911376	0.534255	1.351351	1.602766	0.565682	0.345695	0.219987
		3	2,716	45,064	45,064	0.951127	0	0	0	0	4.312569	2.359012	0	0.073719	2.948765	1.769259	1.179506
		4	3,019	50,094	50,094	0.804617	1	14.256	14.256	14.256	4.111406	1.98939	0	0.066313	2.155172	1.790451	0.397878
		5	3,374	55,986	55,986	1.085757	0	0	0	0	0.652625	0	0.118659	0	0.296648	0.296648	0
		6	3,383	56,131	56,131	0.942062	0	0	0	0	6.360947	1.390533	0	0.059172	1.923077	1.863905	0.088757
		7	4,550	75,513	75,513	1.051418	0	0	0	0	3.232901	0.065978	0.08797	0.043985	1.407521	1.385529	0.021993
		8	4,783	79,384	79,384	1.006736	0	0	0	0	10.564854	1.108787	0.125523	0.041841	4.037657	3.682008	0.355649
		Final	8,679	144,073	144,073	0.85784	1	5.151589	5.151589	5.151589	7.872291	3.365606	0.069156	0.092208	3.41171	2.627939	0.806823
	2	1	2,234	37,060	37,060	0.968248	0	0	0	0	15.284626	0.044823	2.868669	0.179292	5.602869	5.558046	0.044823
		2	3,212	53,306	53,306	1.012939	0	0	0	0	11.24961	0	0.062325	0.124649	3.17856	3.17856	0
		3	2,465	40,898	40,898	0.941707	0	0	0	0	11.33225	0	18.31844	0	5.483347	5.483347	0
		4	2,830	46,965	46,965	0.832729	0	0	0	0	10.187478	1.945525	21.330032	0.070746	5.65971	5.55359	0.10612
		5	3,418	56,726	56,726	1.033104	0	0	0	0	4.802343	0	19.355783	0.292826	2.254758	2.254758	0
		6	4,269	70,847	70,847	0.785084	0	0	0	0	15.307079	3.258322	19.643694	0.187529	5.227379	4.758556	0.492264
		7	4,722	78,389	78,389	1.01993	0	0	0	0	6.781098	0	19.029455	0.381437	2.881967	2.881967	0
		8	5,297	87,937	87,937	0.889511	0	0	0	0	13.449188	5.421232	19.531545	15.111447	5.817907	4.552323	1.435587
		Final	10,718	177,950	177,950	0.7127	6	10.749059	3.432288	20.593725	13.093794	4.965002	2.790481	11.012599	5.30098	4.489034	0.933271

4	**1**	1	2,064	34,253	34,253	1.04436	0	0	0	0	0.485201	0.14556	0.09704	0.291121	0.194081	0.14556	0.04852
		2	3,158	52,402	52,402	1.047637	0	0	0	0	2.155309	1.204437	0.063391	0	1.458003	1.014263	0.602219
		3	2,620	43,473	43,473	1.053035	0	0	0	0	1.490256	0.267482	0	0.076423	0.878869	0.764234	0.114635
		4	7,476	124,117	124,117	0.720831	1	10.809467	10.809467	10.809467	5.700522	9.367055	0.026763	0.26763	5.165262	1.632544	3.586244
		5	6,700	111,198	111,198	0.889071	1	4.71019	4.71019	4.71019	5.106764	6.286397	0.238913	0.298641	5.136628	3.076004	2.180081
		6	4,070	67,551	67,551	0.991187	1	6.598118	6.598118	6.598118	2.139169	2.434227	0.147529	0.147529	0.983526	0.565527	0.417999
		7	5,625	93,375	93,375	0.973235	0	0	0	0	2.258983	4.144433	0.071149	0.071149	1.956599	0.960512	0.996087
		8	4,791	79,520	79,520	1.051837	0	0	0	0	2.360067	0.062657	0.083542	0.125313	0.93985	0.93985	0
		Final	8,083	134,174	134,174	0.93948	2	17.827412	9.668647	19.337294	2.660891	2.537129	0.09901	0.173267	1.658416	0.977723	0.680693
	2	1	2,155	35,746	35,746	1.000956	0	0	0	0	2.04461	0.836431	2.788104	4.182156	0.975836	0.789963	0.185874
		2	3,701	61,423	61,423	0.962695	1	3.63	3.63	3.63	4.840454	4.164413	0.108167	1.189832	2.109248	1.676582	0.459708
		3	2,946	48,878	48,878	0.873903	0	0	0	0	5.878355	5.028882	0.135916	20.931023	3.19402	2.616378	0.577642
		4	4,790	79,504	79,504	0.661346	1	22.52706	22.52706	22.52706	5.932734	6.371423	0.208899	19.072488	4.386881	2.40234	2.026321
		5	3,524	58,481	58,481	1.02802	0	0	0	0	0.511218	0.880432	0.170406	19.965919	0.426015	0.227208	0.198807
		6	3,604	59,817	59,817	0.851426	0	0	0	0	4.109969	5.470703	0.05554	15.606776	2.804776	1.832824	0.971952
		7	5,481	90,978	90,978	0.95793	0	0	0	0	1.277839	3.742242	0.730194	17.761957	1.515152	0.511135	1.004016
		8	8,222	136,492	136,492	0.755981	4	33.822117	11.623725	46.494899	5.462952	7.421827	13.663463	10.743399	6.338971	2.396885	3.954252
		Final	11,001	182,651	182,651	0.752122	9	7.102235	3.629415	32.664734	6.337516	7.27405	0.163666	15.975632	3.409711	2.109474	1.363884

5	2	1	32,098	532,968	528,148	0.629661	11	31.165186	8.187613	90.06374	7.80807	8.144571	1.557875	7.188036	3.181181	2.143636	1.087397
		2	3,559	59,065	59,065	1.000813	0	0	0	0	3.543307	0	0.056243	20.556805	1.546682	1.546682	0
		3	2,533	42,027	42,027	0.987671	0	0	0	0	4.980237	0	0.079051	19.051383	2.134387	2.134387	0
		4	4,482	74,395	74,395	0.658639	3	9.865812	5.480031	16.440092	4.956463	1.607502	0.178611	22.638982	1.987051	1.719134	0.267917
		5	3,735	61,990	61,990	1.002219	0	0	0	0	7.717042	0	0.053591	25.482315	3.644159	3.644159	0
		6	3,136	52,045	52,045	0.966496	0	0	0	0	6.702841	0	0.127673	26.172997	2.42579	2.42579	0
		7	4,435	73,615	73,615	1.021746	0	0	0	0	6.656137	0	0.451264	26.827617	3.249097	3.249097	0
		8	5,409	89,787	89,787	1.002811	1	3.225751	3.225751	3.225751	8.26859	0	14.576397	25.749168	2.811691	2.811691	0
		Final	13,835	229,712	229,712	0.742935	14	24.505688	7.796172	109.146402	4.012435	1.937536	17.582418	18.052342	1.12059	0.93262	0.18797
	1	1	2,085	34,585	34,585	1.035519	0	0	0	0	1.344861	0	0	0.096061	0.480307	0.480307	0
		2	2,995	49,696	49,696	1.065239	0	0	0	0	1.537433	0.033422	0.066845	0.066845	0.735294	0.735294	0
		3	2,272	37,688	37,688	1.034897	0	0	0	0	2.600264	0	0	0.088145	1.101807	1.101807	0
		4	3,398	56,384	56,384	0.872649	4	12.642764	5.196583	20.786332	1.531664	0.736377	0.05891	0.05891	0.883652	0.530191	0.353461
		5	3,309	54,901	54,901	1.051355	0	0	0	0	1.724138	0	0.181488	0.30248	0.816697	0.816697	0
		6	2,949	48,933	48,933	1.015214	0	0	0	0	1.866938	0.644942	0	0.203666	0.814664	0.712831	0.101833
		7	4,334	71,927	71,927	1.043943	1	1.416059	1.416059	1.416059	2.586008	0.36943	0	0.046179	1.200647	1.015932	0.184715
		8	4,361	72,374	72,374	1.063682	0	0	0	0	0.87196	0.367141	0.18357	0.091785	0.390087	0.275356	0.114732
		Final	7,905	131,224	131,224	0.970488	9	22.951624	5.267529	47.407765	1.986839	0.847887	0.02531	0.02531	0.974437	0.771956	0.20248

6	**2**	1	2,253	37,389	34,774	0.955242	0	0	0	0	0.133333	0.311111	3.422222	5.688889	0.044444	0.044444	0
		2	3,931	65,242	56,907	0.892356	0	0	0	0	3.029532	1.985743	0.738289	0.86558	1.45112	1.247454	0.203666
		3	3,319	55,084	48,361	0.695407	0	0	0	0	3.196622	1.025332	5.669481	7.38842	1.568154	1.296743	0.301568
		4	3,629	60,229	60,229	0.670021	1	4.939874	4.939874	4.939874	7.032543	4.495312	15.250965	11.445119	4.219526	3.116382	1.130723
		5	4,205	69,792	64,211	0.889507	2	13.043298	8.505589	17.011179	2.927178	2.594003	0.999524	5.188006	1.45169	1.166111	0.285578
		6	3,361	55,784	54,655	0.906683	0	0	0	0	4.109589	2.650387	18.58249	1.191185	2.293032	1.459202	0.83383
		7	5,285	87,723	87,723	0.949812	0	0	0	0	5.168497	2.461189	17.588035	9.485044	2.764105	2.120409	0.643696
		8	5,326	88,411	88,411	0.917297	1	2.920943	2.920943	2.920943	3.569416	2.799173	21.360135	19.086981	1.87864	1.502912	0.375728
		Final	15,210	252,534	240,446	0.513432	5	8.347312	3.710474	18.55237	5.753929	4.787269	8.187019	13.783126	2.965739	2.43309	0.545801
	1	1	2,277	37,769	36,126	0.970548	0	0	0	0	0.615655	0.615655	2.638522	2.242744	0.395778	0.307828	0.087951
		2	3,482	57,782	57,782	1.005973	0	0	0	0	2.443231	3.190572	0.057488	0.114976	1.121012	0.948548	0.201207
		3	5,004	83,055	83,055	0.90153	0	0	0	0	3.819236	3.539292	0.039992	0.119976	2.779444	1.339732	1.439712
		4	3,179	52,748	52,748	0.754021	1	17.588821	17.588821	17.588821	3.967254	2.109572	0.062972	0.188917	1.952141	1.38539	0.566751
		5	3,285	54,509	54,509	1.031587	0	0	0	0	1.767215	1.614869	0.365631	0	1.096892	0.944546	0.152346
		6	3,340	55,423	55,423	0.97066	0	0	0	0	4.195385	2.517231	0	0	1.798022	1.648187	0.179802
		7	4,807	79,778	79,778	1.033584	0	0	0	0	2.560366	1.373855	0.124896	0.083264	1.706911	1.248959	0.478768
		8	5,167	85,757	85,757	0.997837	0	0	0	0	3.931061	1.065066	0	0.077459	1.897754	1.491092	0.426026
		Final	10,083	167,373	167,373	0.774672	5	6.90525	5.152457	25.762285	6.994048	4.980159	0.039683	0.079365	3.849206	2.787698	1.150794

7	2	1	2,404	39,903	38,012	0.916633	0	0	0	0	5.122865	4.539775	22.115785	2.207414	1.957518	1.749271	0.249896
		2	4,973	82,545	82,545	0.788953	2	17.468811	11.014611	22.029222	7.625755	5.895372	21.267606	19.979879	4.225352	3.199195	1.066398
		3	2,839	47,108	47,108	0.896389	0	0	0	0	4.337094	6.523272	19.111425	19.287729	3.808181	1.974612	1.833568
		4	3,318	55,070	55,070	0.679831	2	16.843174	9.135469	18.270939	5.942685	7.209653	15.71644	16.923077	3.951735	2.533937	1.538462
		5	3,686	61,177	61,177	0.925798	0	0	0	0	2.769481	5.077383	19.657888	19.54928	2.090687	1.140375	1.004616
		6	3,610	59,907	59,907	0.858963	1	8.341294	8.341294	8.341294	5.849737	8.622124	18.186859	12.337122	3.853618	2.273357	1.607984
		7	5,140	85,324	85,324	0.874316	1	1.191314	1.191314	1.191314	5.275453	5.917851	16.410356	17.792486	3.73759	2.316527	1.440529
		8	5,374	89,200	89,200	0.855855	0	0	0	0	5.49246	4.85943	17.464159	19.679762	3.127909	2.178365	0.968162
		Final	9,786	162,469	162,469	0.741989	1	7.100625	7.100625	7.100625	6.920168	8.146785	18.409486	12.450169	4.047838	2.412348	1.676377
	1	1	2,163	35,887	35,240	0.994113	0	0	0	0	1.342593	1.111111	3.148148	1.62037	0.740741	0.462963	0.277778
		2	3,200	53,104	53,104	1.013081	0	0	0	0	2.752581	3.440726	0.375352	0	1.876759	0.969659	0.9071
		3	2,482	41,180	41,180	0.975862	0	0	0	0	3.106091	3.872529	0.080678	0	2.46067	1.331182	1.169827
		4	2,885	47,869	47,869	0.779704	1	6.132882	6.132882	6.132882	3.851492	5.898681	0.138793	0.069396	2.949341	1.526718	1.457321
		5	3,344	55,497	55,497	1.050773	0	0	0	0	0.77821	2.394493	0.119725	0.059862	1.107453	0.359174	0.748279
		6	3,376	56,031	56,031	0.97092	0	0	0	0	2.875778	4.684257	0.118589	0.059294	2.43107	1.185888	1.245182
		7	4,805	79,756	79,756	1.035332	0	0	0	0	2.019992	3.540192	0.041649	0.083299	2.14494	0.93711	1.270304
		8	4,978	82,635	82,635	1.039663	0	0	0	0	3.155779	2.894472	0.120603	0.040201	1.768844	1.145729	0.643216
		Final	8,371	138,964	138,964	0.894458	3	8.042687	5.015993	15.04798	5.102772	4.421606	0.119503	0.047801	2.413958	1.637189	0.776769

8	1	1	2,119	35,153	35,153	1.028942	0	0	0	0	1.181474	0.567108	0	0.189036	0.47259	0.283554	0.189036
		2	3,034	50,341	50,341	1.073595	0	0	0	0	0.626856	0.13197	0.13197	0.065985	0.263939	0.197954	0.065985
		3	2,857	47,409	47,409	0.921088	0	0	0	0	2.17239	1.366503	0	0.140154	1.121233	0.946041	0.175193
		4	3,570	59,248	59,248	0.851491	3	11.814186	5.909803	17.729408	2.102607	1.289599	0	0.05607	0.504626	0.504626	0
		5	3,214	53,337	53,337	1.069489	0	0	0	0	1.05886	0.031143	0.062286	0.124572	0.560573	0.560573	0
		6	3,027	50,230	50,230	1.03995	0	0	0	0	1.455026	0.82672	0.26455	0	0.496032	0.429894	0.066138
		7	4,186	69,479	69,479	1.077346	0	0	0	0	0.765001	0.741095	0.143438	0.143438	0.597657	0.406407	0.19125
		8	4,259	70,690	70,690	1.057776	0	0	0	0	2.067669	0.37594	0.046992	0.046992	0.93985	0.75188	0.18797
		Final	7,923	131,679	131,679	0.898272	4	7.417624	4.468535	17.874141	3.813131	2.727273	0.10101	0.025253	1.919192	1.464646	0.479798
	2	1	2,354	39,051	37,125	0.918016	0	0	0	0	7.486176	0.595491	20.374309	3.487877	3.572948	3.402807	0.212675
		2	3,486	57,861	57,861	0.958885	0	0	0	0	5.713465	1.320701	16.881998	0.97617	3.100775	2.756245	0.344531
		3	2,307	38,279	36,635	0.966959	0	0	0	0	1.649306	0.651042	3.515625	4.036458	1.649306	1.041667	0.651042
		4	2,701	44,818	42,742	0.860629	0	0	0	0	2.89103	2.446256	2.853966	2.668643	3.113417	1.779096	1.704967
		5	3,271	54,281	54,281	1.04432	0	0	0	0	3.916769	1.009792	6.364749	0.489596	1.713586	1.652387	0.0612
		6	5,118	84,953	83,275	0.620961	0	0	0	0	3.264907	1.70088	10.048876	8.054741	1.290323	0.957967	0.371457
		7	6,870	114,036	110,798	0.825473	0	0	0	0	4.863842	1.572739	3.771662	3.655162	1.587302	1.383428	0.218436
		8	9,061	150,434	146,150	0.619318	3	10.097064	6.211391	18.634172	5.939501	1.644955	6.2155	6.767498	2.649592	2.252153	0.397439
		Final	8,863	147,140	141,212	0.798958	1	24.686588	24.686588	24.686588	6.331828	1.783296	15.902935	3.837472	2.68623	2.370203	0.327314

9	2	1	2,387	39,605	35,646	0.900945	0	0	0	0	0.083893	0	1.006711	1.090604	0	0	0
		2	3,037	50,392	50,392	1.054836	0	0	0	0	1.450231	0	0.06592	0	0.659196	0.659196	0
		3	2,382	39,522	39,522	1.028902	0	0	0	0	2.35393	0	0	0.168138	1.134931	1.134931	0
		4	3,163	52,485	52,485	0.818673	1	8.119124	8.119124	8.119124	2.278481	2.310127	0.063291	0.189873	1.993671	0.917722	1.075949
		5	3,448	57,222	57,222	1.060632	0	0	0	0	0.493469	0.232221	0.058055	0	0.377358	0.261248	0.11611
		6	3,431	56,938	56,938	1.023965	0	0	0	0	1.487748	0.933489	0.116686	0	0.700117	0.437573	0.262544
		7	4,483	74,404	74,404	1.075997	0	0	0	0	0.870536	0.357143	0.133929	0.089286	0.491071	0.3125	0.178571
		8	5,023	83,377	83,377	1.05928	0	0	0	0	0.936255	1.075697	0.119522	0.119522	0.677291	0.338645	0.338645
		Final	8,517	141,394	141,394	0.945123	3	23.334251	9.421918	28.265755	2.795396	2.008457	0.211416	0.093963	1.080573	0.669486	0.422833
	1	1	12,173	202,095	199,688	0.428114	9	14.478176	4.518505	40.666546	12.958094	5.201315	13.344289	14.757601	3.516845	2.826623	0.690222
		2	3,728	61,870	61,870	0.854004	0	0	0	0	10.174497	0	16.885906	7.516779	4.456376	4.456376	0
		3	2,536	42,080	42,080	0.886494	0	0	0	0	8.645874	0	13.817608	0	4.382156	4.382156	0
		4	3,832	63,594	63,594	0.686787	2	7.871812	6.761317	13.522635	5.797858	0.992426	14.442413	22.355706	2.454949	2.454949	0
		5	3,432	56,954	56,954	1.014106	0	0	0	0	4.316127	0	4.986877	5.511811	1.983085	1.983085	0
		6	4,099	68,027	68,027	0.860757	2	14.83988	7.945593	15.891187	6.225586	0	18.286133	0.048828	2.270508	2.270508	0
		7	6,611	109,733	109,733	0.884343	2	10.729646	7.631531	15.263062	4.479419	0.514528	19.249395	5.432809	1.588983	1.467918	0.121065
		8	5,137	85,263	85,263	0.959759	0	0	0	0	3.622906	0.311648	18.134009	20.412933	1.733541	1.538761	0.19478
		Final	11,352	188,467	188,467	0.623756	8	17.86475	6.638314	53.10651	5.066526	2.775575	15.754692	14.750198	2.669839	2.149969	0.51987

10	**1**	1	2,210	36,658	35,696	0.980654	0	0	0	0	0.589035	0.226552	3.081106	2.718623	0.226552	0.226552	0
		2	3,057	50,724	50,724	1.081411	0	0	0	0	0.294695	0.032744	0	0	0.196464	0.16372	0.032744
		3	2,405	39,897	39,897	0.904087	0	0	0	0	1.998335	2.53955	0.083264	0	1.998335	1.290591	0.707744
		4	3,193	52,982	52,982	0.761661	1	8.923882	8.923882	8.923882	6.77116	4.639498	0.062696	0.062696	2.821317	2.225705	1.065831
		5	3,356	55,685	55,685	1.059406	0	0	0	0	0.268416	0.477185	0.178944	0.059648	0.29824	0.119296	0.178944
		6	3,511	58,264	58,264	1.012935	0	0	0	0	2.993158	2.67959	0.057013	0	1.68187	1.083238	0.684151
		7	4,411	73,203	73,203	1.081968	0	0	0	0	0.748639	0.476407	0.045372	0.22686	0.476407	0.34029	0.136116
		8	4,889	81,345	81,345	0.995352	0	0	0	0	2.067131	1.964797	0.081867	0.081867	1.616865	0.818666	0.818666
		Final	9,363	155,425	151,108	0.829755	4	28.33956	9.842683	39.370733	5.395299	5.90812	1.07906	1.239316	2.435897	1.655983	0.801282
	2	1	2,125	35,250	35,250	1.013811	0	0	0	0	2.073516	0.329877	0.094251	0.942507	0.801131	0.659755	0.141376
		2	3,213	53,314	53,314	1.014628	0	0	0	0	4.018692	0.965732	0.186916	17.009346	1.993769	1.713396	0.280374
		3	2,378	39,448	39,448	0.927758	0	0	0	0	4.463158	2.989474	0.336842	18.736842	3.242105	1.810526	1.431579
		4	3,707	61,513	61,513	0.650865	1	1.988437	1.988437	1.988437	5.858531	4.103672	3.158747	13.714903	3.320734	2.078834	1.241901
		5	3,458	57,380	57,380	1.044365	0	0	0	0	2.575977	0.144718	0	13.892909	1.30246	1.21563	0.086831
		6	4,513	74,895	74,895	0.852716	1	14.536186	14.536186	14.536186	7.649667	4.235033	0.133038	21.374723	4.523282	3.237251	1.441242
		7	4,482	74,383	74,383	1.016304	0	0	0	0	3.014066	0.223264	0.044653	12.32418	1.49587	1.473543	0.022326
		8	4,704	78,065	78,065	0.99969	0	0	0	0	3.637524	1.063603	16.379494	16.54967	1.595405	1.425229	0.170177
		Final	9,241	153,392	148,179	0.797103	2	3.142688	2.75458	5.509159	8.486685	5.098506	1.385581	15.436242	4.351591	3.290756	1.223208

11	2	1	33,918	563,132	563,132	0.546116	6	11.385233	5.130139	30.780834	17.611676	8.055433	11.43152	12.767212	9.394073	7.00575	2.485626
	1	1	2,157	35,779	35,779	1.007228	0	0	0	0	2.924791	0	0.185701	0.232126	1.021356	1.021356	0
		2	3,165	52,520	52,520	1.003253	0	0	0	0	2.909551	0	0	0.221379	0.980392	0.980392	0
		3	2,400	39,817	39,817	0.977588	0	0	0	0	2.962036	0	0.250313	0.417188	1.335002	1.335002	0
		4	4,095	67,958	67,958	0.75466	2	8.461354	5.620589	11.241178	8.504399	0.171065	0.097752	0.415445	2.8348	2.785924	0.048876
		5	3,781	62,745	62,745	1.016364	0	0	0	0	5.240868	0.211752	0.105876	0.105876	1.932239	1.90577	0.026469
		6	4,162	69,074	69,074	0.89776	1	3.053882	3.053882	3.053882	3.847079	0.192354	0.096177	0.96177	1.154124	1.154124	0
		7	5,743	95,318	95,318	1.002216	1	11.832002	11.832002	11.832002	5.383275	0.121951	0.958188	0.662021	2.003484	2.003484	0
		8	15,360	255,001	255,001	0.657304	13	30.605648	12.184881	158.403451	8.419613	0.013023	20.765775	0.175816	0.651169	0.651169	0

Appendix E – General Statistical Data (referent to the user tests)

Units			unit	msec	msec	m/s	unit	m/s^2	m/s^2	m/s^2	%	%	%	%	%	%	%
User	Commands Layout	Map	Entries	Time to Complete	Time to Complete (Effective)	Average Speed	Number of Crashes	Most Violent Crash	Average Crash Acceleration	Crash Acceleration Sum	Front Direction Command Usage	Back Direction Command Usage	Front Speed Command Usage	Back Speed Command Usage	Time Spent Compensating	Front Compensation	Back Compensation
General Statistics																	
			Sum				**Average**	**Total**	**Max**	**Average**	**Sum**	**Average**					
			866,256	14,378,455	14,292,345		0.92	208	39.56	7.38	1585.30	3.91	2.06	4.65	5.38	1.98	1.51	0.49
				Mean Time Per User	**Mean Time to Start**			**Per User**				**Max**	**Max**	**Max**	**Max**	**Max**	**Max**	**Max**
				1,437,846	478.39			20.80				15.31	9.37	22.27	26.83	6.34	5.56	3.95
Control Layouts Comparison																		
				Sum			**Average**	**Sum**	**Max**		**Sum**	**Average**						
				Layout 1							**Layout 1**							
				6,466,631	6,453,649		0.96	83	38.51		574.27	2.92	1.71	1.74	1.25	1.53	1.12	0.42
				Layout 2							**Layout 2**							
				7,911,824	7,838,696		0.89	125	39.56		1011.03	4.90	2.40	7.55	9.51	2.43	1.89	0.56

Appendix F – General Statistical Data (referent to the user tests during the final map)

Unit	msec	m/s	unit	m/s^2	m/s^2	m/s^2	%	%	%	%	%	%	%
Commands Layout	Time to Complete (Effective)	Average Speed	Number of Crashes	Most Violent Crash	Average Crash Acceleration	Crash Acceleration Sum	Front Direction Command Usage	Back Direction Command Usage	Front Speed Command Usage	Back Speed Command Usage	Time Spent Compensating	Front Compensation	Back Compensation
1	162034	0.755673	7	13.174626	5.766285	40.363996	5.083009	6.579217	0.10248	0.081984	3.863497	2.275056	1.598688
	127718	0.959301	5	38.507759	17.857364	89.286822	2.015343	0	0.780133	0.104018	0.520088	0.520088	0
	144073	0.85784	1	5.151589	5.151589	5.151589	7.872291	3.365606	0.069156	0.092208	3.41171	2.627939	0.806823
	134174	0.93948	2	17.827412	9.668647	19.337294	2.660891	2.537129	0.09901	0.173267	1.658416	0.977723	0.680693
	131224	0.970488	9	22.951624	5.267529	47.407765	1.986839	0.847887	0.02531	0.02531	0.974437	0.771956	0.20248
	167373	0.774672	5	6.90525	5.152457	25.762285	6.994048	4.980159	0.039683	0.079365	3.849206	2.787698	1.150794
	138964	0.894458	3	8.042687	5.015993	15.04798	5.102772	4.421606	0.119503	0.047801	2.413958	1.637189	0.776769
	131679	0.898272	4	7.417624	4.468535	17.874141	3.813131	2.727273	0.10101	0.025253	1.919192	1.464646	0.479798
	188467	0.623756	8	17.86475	6.638314	53.10651	5.066526	2.775575	15.754692	14.750198	2.669839	2.149969	0.51987
	151108	0.829755	4	28.33956	9.842683	39.370733	5.395299	5.90812	1.07906	1.239316	2.435897	1.655983	0.801282

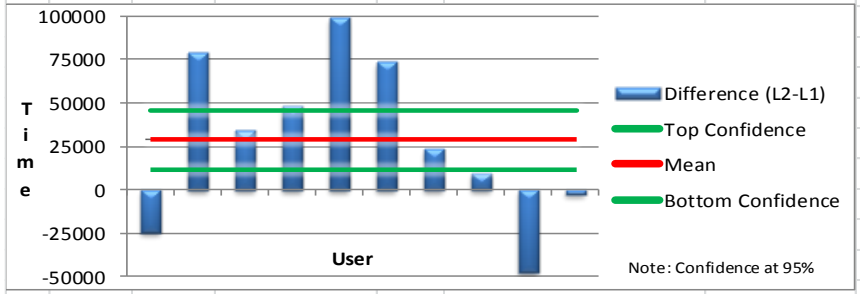
2	137335	0.823836	3	4.640312	2.298884	6.896652	4.860943	4.292624	3.482467	22.623942	2.394196	1.789601	0.616687
	206202	0.792791	11	39.562374	17.762192	195.384109	2.544693	0	19.922693	10.992108	0.885811	0.885811	0
	177950	0.7127	6	10.749059	3.432288	20.593725	13.093794	4.965002	2.790481	11.012599	5.30098	4.489034	0.933271
	182651	0.752122	9	7.102235	3.629415	32.664734	6.337516	7.27405	0.163666	15.975632	3.409711	2.109474	1.363884
	229712	0.742935	14	24.505688	7.796172	109.146402	4.012435	1.937536	17.582418	18.052342	1.12059	0.93262	0.18797
	240446	0.513432	5	8.347312	3.710474	18.55237	5.753929	4.787269	8.187019	13.783126	2.965739	2.43309	0.545801
	162469	0.741989	1	7.100625	7.100625	7.100625	6.920168	8.146785	18.409486	12.450169	4.047838	2.412348	1.676377
	141212	0.798958	1	24.686588	24.686588	24.686588	6.331828	1.783296	15.902935	3.837472	2.68623	2.370203	0.327314
	141394	0.945123	3	23.334251	9.421918	28.265755	2.795396	2.008457	0.211416	0.093963	1.080573	0.669486	0.422833
	148179	0.797103	2	3.142688	2.75458	5.509159	8.486685	5.098506	1.385581	15.436242	4.351591	3.290756	1.223208

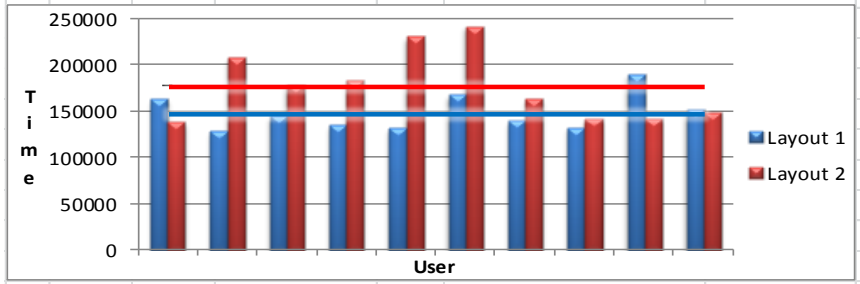
Control Layouts Comparison - Final Map													
	Average												
1	147,681	0.8503695	4.8	16.6182881	7.4829396	35.2709115	4.5990149	3.4142572	1.8170037	1.661872	2.371624	1.6868247	0.7017197
2	176,755	0.7620989	5.5	15.3171132	8.2593136	44.8800119	6.1137387	4.0293525	8.8038162	12.4257595	2.8243259	2.1382423	0.7297345
	Sum		**Sum**	**Max**		**Sum**							
1	1476814		48	38.507759		352.709115							
2	1767550		55	39.562374		448.800119							

Appendix G – Statistical Analysis of the Effective Time to Complete

	L1	L2	Difference (Δ)
Effective Time to Complete	162034	137335	-24699
	127718	206202	78484
	144073	177950	33877
	134174	182651	48477
	131224	229712	98488
	167373	240446	73073
	138964	162469	23505
	131679	141212	9533
	188467	141394	-47073
	151108	148179	-2929
Mean	147681	176755	29073.6

Mean	29073.6
Standard Error	14803.1777
Median	28691
Standard Deviation	46811.75814
Sample Variation	2191340700
Kurtosis	-0.873934434
Skewness	-0.121482101
Range	145561
Minimum	-47073
Maximum	98488
Sum	290736
Count	10
Confidence Level (95%)	33487.11447





T-Tests

Paired, Two samples for means	Layout 1	Layout 2
Mean	147681.4	176755
Variance	385098513.4	1428069236
Observations	10	10
Pearson Correlation	-0.254976358	
Hypothesized Mean Difference	0	
dF	9	
t Stat	-1.964010741	
P(T<=t) one-tail	0.040559851	
t Critical one-tail	1.833112933	
P(T<=t) two-tail	0.081119703	
t Critical two-tail	2.262157163	

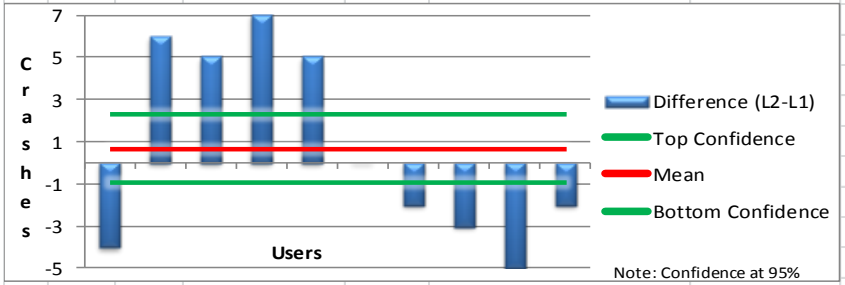
Two sample with unequal variance	*Variável 1*	*Variável 2*
Mean	147681.4	176755
Variance	385098513.4	1428069236
Observations	10	10
Hypothesized Mean Difference	0	
dF	14	
t Stat	-2.159135103	
P(T<=t) one-tail	0.02434056	
t Critical one-tail	1.761310136	
P(T<=t) two-tail	0.04868112	
t Critical two-tail	2.144786688	

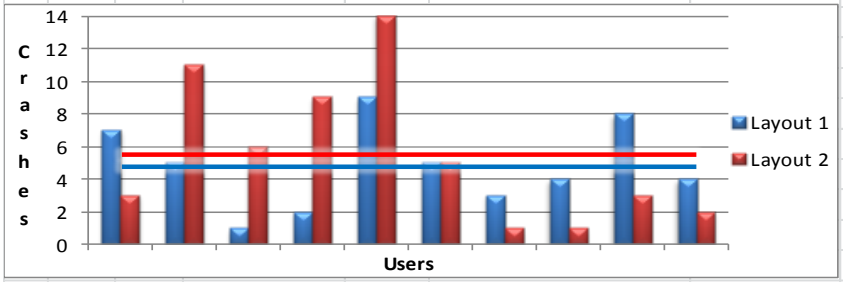
Effective Time to Complete
(during the final map)

Appendix H – Statistical Analysis of the Number of Crashes

	L1	L2	Difference (Δ)
Number of Crashes	7	3	-4
	5	11	6
	1	6	5
	2	9	7
	9	14	5
	5	5	0
	3	1	-2
	4	1	-3
	8	3	-5
	4	2	-2
Mean	4.8	5.5	0.7

Mean	0.7
Standard Error	1.445683229
Median	-1
Standard Deviation	4.57165178
Sample Variation	20.9
Kurtosis	-1.898166469
Skewness	0.259556705
Range	12
Minimum	-5
Maximum	7
Sum	7
Count	10
Confidence Level (95%)	3.270362673





T-Tests

Paired, Two samples for means	Layout 1	Layout 2
Mean	4.8	5.5
Variance	6.622222222	20.05555556
Observations	10	10
Pearson Correlation	0.250675341	
Hypothesized Mean Difference	0	
dF	9	
t Stat	-0.484200125	
P(T<=t) one-tail	0.319904307	
t Critical one-tail	1.833112933	
P(T<=t) two-tail	0.639808614	
t Critical two-tail	2.262157163	

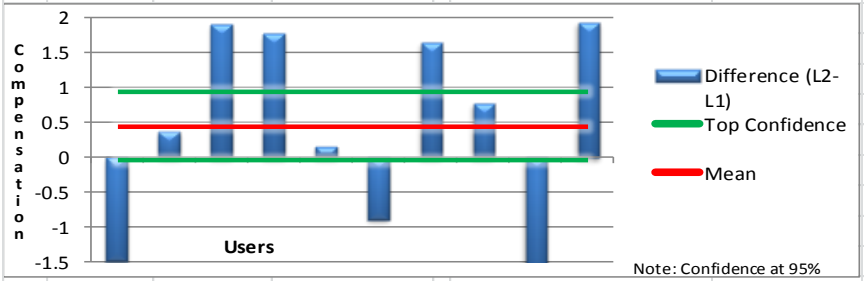
Two sample with unequal variance	Layout 1	Layout 2
Mean	4.8	5.5
Variance	6.622222222	20.05555556
Observations	10	10
Hypothesized Mean Difference	0	
dF	14	
t Stat	-0.428571429	
P(T<=t) one-tail	0.337377891	
t Critical one-tail	1.761310136	
P(T<=t) two-tail	0.674755783	
t Critical two-tail	2.144786688	

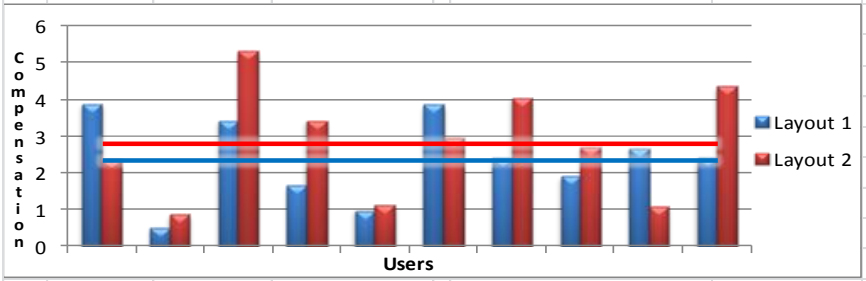
Number of Crashes
(during the final map)

Appendix I – Statistical Analysis of the Time Spent Compensating

	L1	L2	Difference (Δ)
Time Spent Compensating	3.863497	2.394196	-1.469301
	0.520088	0.885811	0.365723
	3.41171	5.30098	1.88927
	1.658416	3.409711	1.751295
	0.974437	1.12059	0.146153
	3.849206	2.965739	-0.883467
	2.413958	4.047838	1.63388
	1.919192	2.68623	0.767038
	2.669839	1.080573	-1.589266
	2.435897	4.351591	1.915694
Mean	2.371624	2.8243259	0.4527019

Mean	0.4527019
Standard Error	0.436336003
Median	0.5663805
Standard Deviation	1.379815594
Sample Variation	1.903891073
Kurtosis	-1.457053302
Skewness	-0.426361157
Range	3.50496
Minimum	-1.589266
Maximum	1.915694
Sum	4.527019
Count	10
Confidence Level (95%)	0.987060614





T-Tests

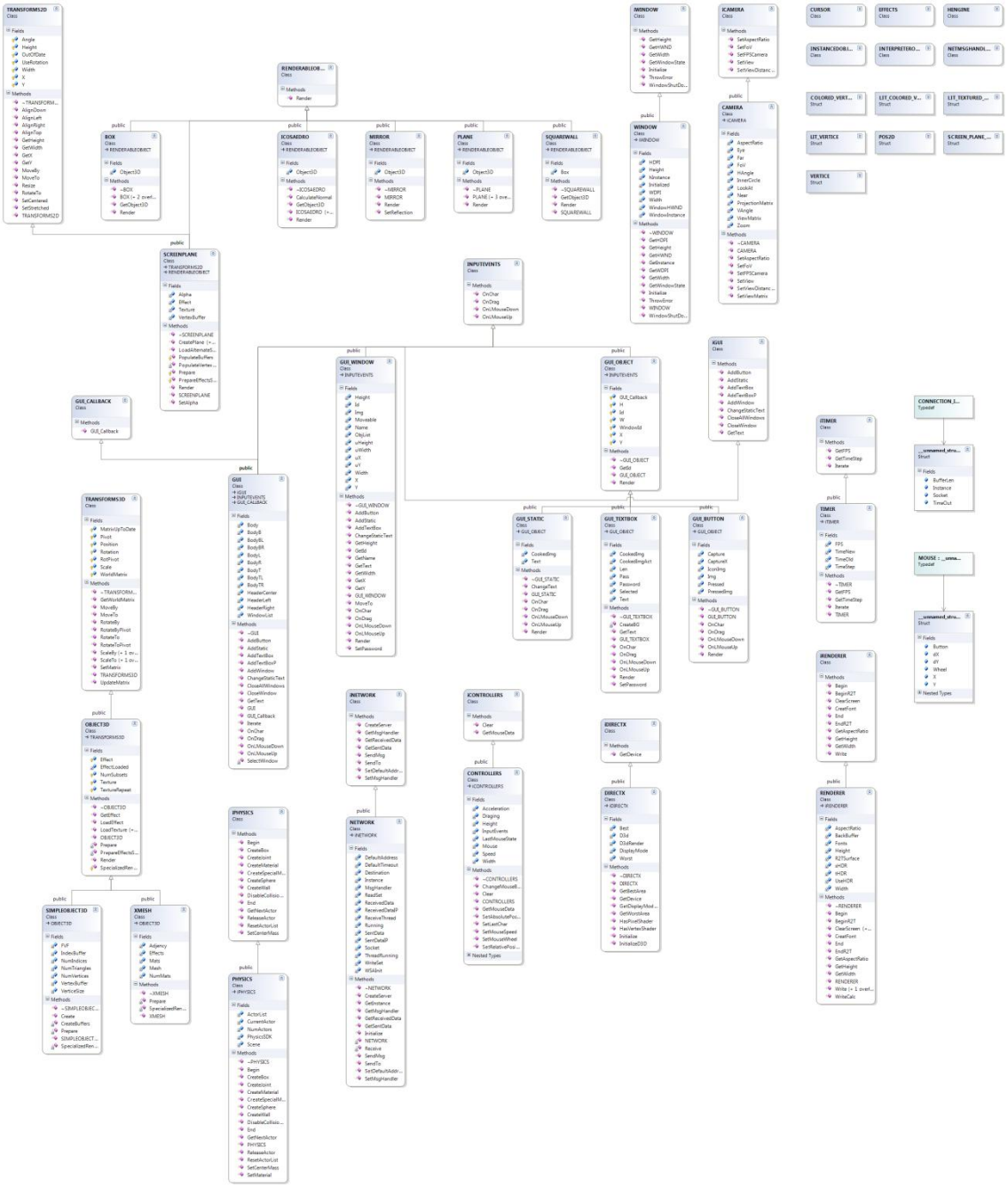
Paired, Two samples for means	Layout 1	Layout 2
Mean	2.371624	2.8243259
Variance	1.297693924	2.247924653
Observations	10	10
Pearson Correlation	0.4806114	
Hypothesized Mean Difference	0	
dF	9	
t Stat	-1.037507556	
P(T<=t) one-tail	0.163284798	
t Critical one-tail	1.833112933	
P(T<=t) two-tail	0.326569596	
t Critical two-tail	2.262157163	

Two sample with unequal variance	Layout 1	Layout 2
Mean	2.371624	2.8243259
Variance	1.297693924	2.247924653
Observations	10	10
Hypothesized Mean Difference	0	
dF	17	
t Stat	-0.760267298	
P(T<=t) one-tail	0.228755089	
t Critical one-tail	1.739606726	
P(T<=t) two-tail	0.457510179	
t Critical two-tail	2.109815578	

Time Spent Compensating
(during the final map)

Appendix J – Class Diagram of the Engine

- Please refer to the CD to view a full-resolution image.



Appendix K – Class Diagram of the Simulation