Serious Game Interaction Techniques Applied to an Operational Satellite Simulator

Christopher Shneider Cerqueira[1], Walter Abrahão dos Santos[2] and Ana Maria Ambrosio[1]
Engineering and Space Technology[1], Applied Computing Laboratory[2]
National Institute for Space Research - INPE
São José dos Campos - Brazil
Email: christophercerqueira@gmail.com, walter.abrahao@lac.inpe.br, ana.ambrosio@inpe.br

***Abstract*—This work describes an initiative to a serious game-like user interface used for an operational satellite simulator. The simulator is part of a simulation architecture intended to provide a richer behavior analysis tool set to the real artefact, as a source of possible behaviors scenarios. In order to interact with the underlying behaviour, the user interface can benefit from: game interaction techniques, increasing cognitive response and, fast data recovery. This is done by extending traditional manipulation techniques and allowing the transition from menus, tables and charts components to data in three-dimensional models, cone-trees, multiple views, mini-maps, touching interface, as well as augmented reality. Finally, a short survey collects final users' satisfaction to this new concept.**

***Keywords*—User interface, simulation, games**

I. Introduction

Simulation is a key element for supporting a wide range of engineering and operational activities during the life cycle and it is recognized as a good practice to generate products [1].

Simulation has many uses, such as training, testing, scientific research, manufacturing, engineering, military and entertainment. In Space programs, the European Space Agency (ESA) [1] indicates that simulation has potential uses across the life cycle, in activities such as:

- Analysis, definition and validation of systems and technical requirements.
- Design validation of high-level performance requirements from various points of view as : electrical, thermal, mechanical, operational.
- Software verification and validation.
- Development of EGSE (Electronic Ground Support Equipment) and test procedures.
- Support of units and subsystem tests activities.
- Prediction of systems performance.
- Development and validation of Operations procedure.
- Troubleshooting for Systems failures and anomalies.
- Control center and crew operator training.

In Brazil, space missions are mainly performed by the National Institute for Space Research (INPE) and they play a key role for its vast territory in water, fishery, agricultural and deforestation monitoring as well as weather/climate data gathering from ground sensing data platforms or obtained from images taken by artificial satellites [2].

Several activities in satellite simulation are performed at INPE, such as those in the Satellite Control Center, which is a key part of the Ground Segment, as shown in Figure 1.

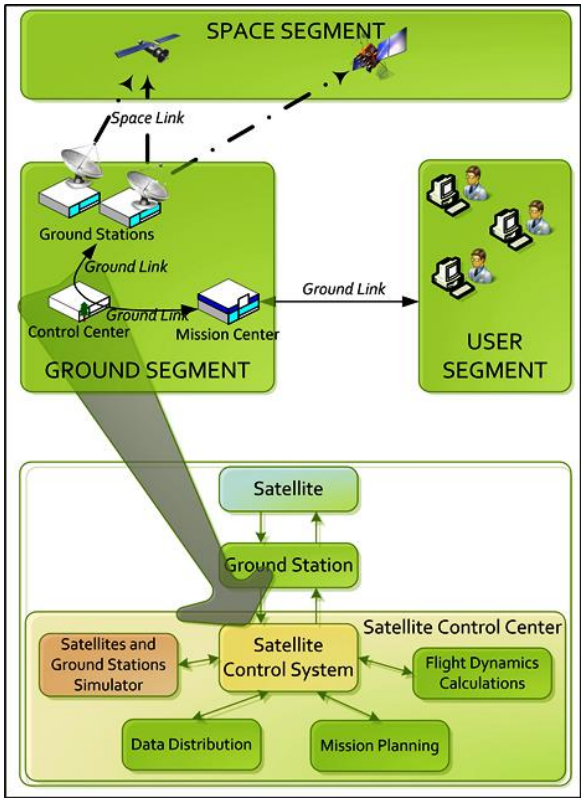

Fig. 1. Space and the Ground Segment facilities, adapted from [3].

A satellite simulation has several stakeholders with different backgrounds that use the simulated data to acquire information, as shown in Figure 2. An example is a faulty satellite condition, which requires an inquiry to the problem by testing conditions and understanding its real behaviour. However, often during operation time, systems engineers and managers who built the satellite are in different projects, and they need to quickly re-learn the possible operational conditions in order to fix the problem. Hence, they can think on possible correction procedures with a final validation by the simulator. The satellite control team receives the engineers'

procedures, which test their communication protocols, just before finally sending the commands to normal operation.

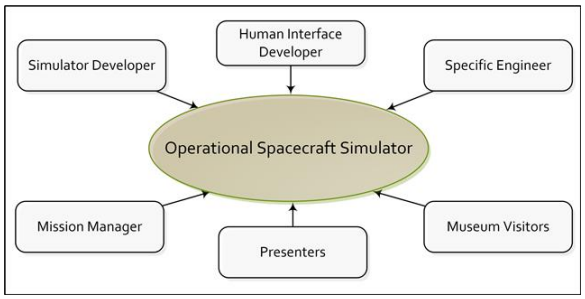

Fig. 2. Key stakeholders for satellite simulations.

The simulation controls the data as it flows within a system, representing the behaviour of a real system, process or phenomenon. This simulation can have many natures, such as: mental, physical, electrical, computational, etc. Computational simulations have wide acceptability as it drastically reduces the systems cost, and is efficient to study some expected scenario [4].

Just like any a computer system, a computation simulation needs a User Interface (UI), categorized by its generation of interaction [5]:

- Generation 1: Typed commands;
- Generation 2: Windows, icons and buttons;
- Generation 3: 3D systems and natural manipulation.

It is worth mentioning that these generations are not time spaced generations but rather how interactions are done. All three generations may co-exist in a time frame; However, they look into different uses and media [6]. The time circumstance of these generations only reflects the requirements of computational power, and cultural factors, desired to run the software [7].

The relation between some relevant components of user interfaces versus the correspondent generation is shown in Figure 3. According to the manipulation abstractness, systems may be driven from specific to more problem-oriented commands. The computational power varies with the effort to understand the interaction. The amount of specific hardware needed may range from dual screen/ keyboard to caves, virtual and augmented reality, gesture recognition. Finally, the system use metaphor requires more computational knowledge as sophistication increases.

Gaming is an industry that traditionally has interaction appeal. The game industry looks at technologies derived from academic research in order to to provide updated gaming sensations. Input methods have evolved: buttons, joysticks, inertial systems and computational vision. Similarly, the output, enables more visual (dimensions, representation, and format), audio (dimension) and haptic (touch, force) feedback [9].

Game interaction strategies are massively by users, the game players, who provide the statistics data about the interaction acceptance. Players complain if something is hard to move, hard to achieve, hard to handle, which ”suggest” new

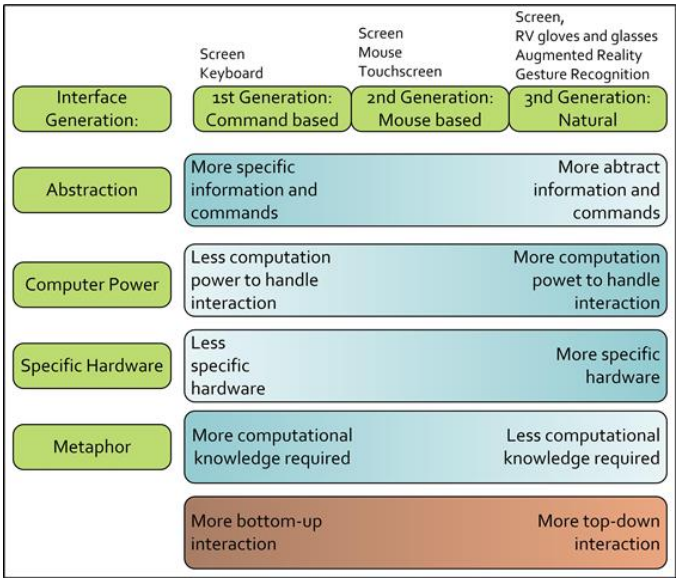

Fig. 3. Classification of interface generation with respect to abstraction, computer power, hardware, metaphor and interaction. Adapted from [8].

ways to manipulate digital content. This all drives requirements to use the same type of interface and more sophisticated methods, as well as, affect the non-game interfaces. This is the case of serious games [10], where they provide characteristics to a successful game-like computer system interaction.

Based on this motivation, this work intends to show a study case on how a transformation process from second generation to third generation interactions affects an operational satellite simulator. It also presents some related initiatives, explains and defines an operational simulation, explores some game characteristics in simulations, correlating second generation items to the proposed UI. Finally, evaluation results are collected among the prospective simulator users (non-programmers, non UI developers).

A. *Related Works*

Each space mission usually builds its own simulation toolset adapted to the mission requirements. Fortunately, there are some attempts to standardize and promote re-usability among the mission phases and/or multiple missions [1].

The main commercial simulator application used for space mission studies is the Analytical Graphics' System Tool Kit (STK)[1], which is a software to model, analyze and visualize space, defense and intelligence systems. The tool provides nice graphics, excellent modeling and screen customization, allows external connection and control as an Active X component or through TCP/IP, integration to Google Earth, Bing, MatLab. Figure 4 shows a visual output example of a satellite scanning operation.

Another approach is Celestia[2], which is a free open source alternative that is mainly focused to astronomy, education and planetarium exploration. It also provides nice graphics, as well as, mouse, keyboard and joystick interactivity. It is also

[1]http://www.agi.com/
[2]http://www.shatters.net/celestia/

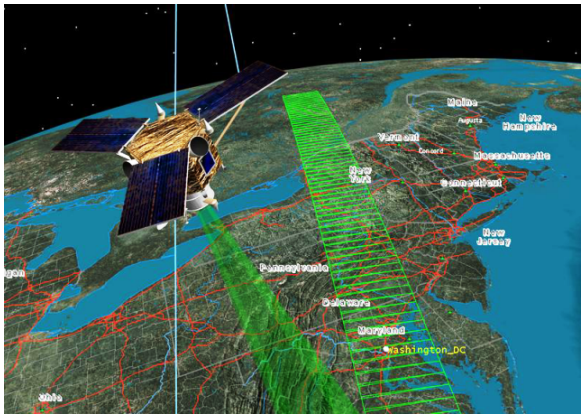

Fig. 4. Example of a System Tool Kit (STK) screen.

expandable, allowing inserting add-ons with more objects and controls, such as new models and gesture recognition. Figure 5 shows an example of the Celestia visual output.

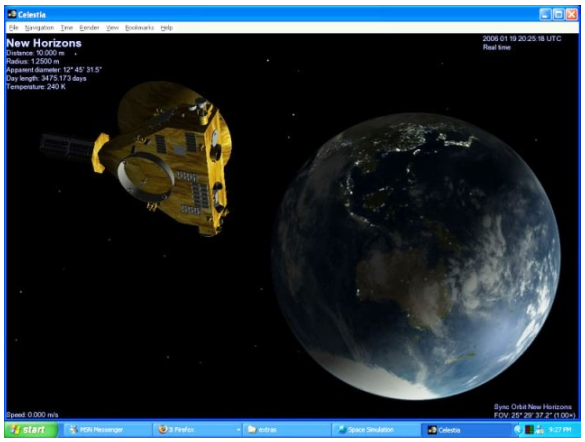

Fig. 5. Example of a Celestia simulation screen.

The VT MÄK[3] from VR Systems, shown at Figure 6, develops distributed simulators based in the communication standard HLA 1.3, HLA 1516, and HLA Evolved [11], which allows local and web distribution with web-services capabilities), and provides connection with Unity[4]. It provides state of the art graphics and game based interactions. It is mainly used to military and war games.

II. Operational Satellite Simulators

In order to create a satellite simulation is necessary to understand the nomenclature of its components, as multiples sources are available. Additionally, it needs contextualization to the INPE's point of view., hence, it follows some definitions:

A simulation is a method for implementing a model over time.

This work simulated scenario represents a group of one satellite and three Earth Stations. The satellite has a group of subsystems (SS) represented by a model, which has several equipment pieces (EQ).

[3]http://www.mak.com
[4]http://unity3d.com/

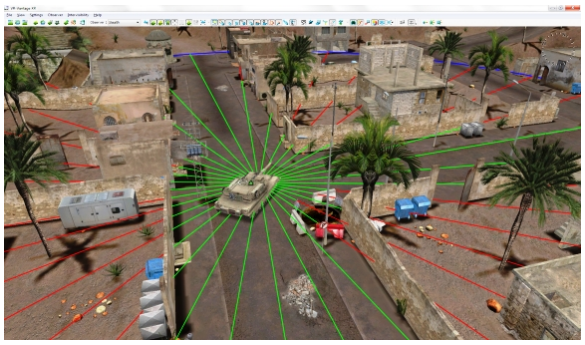

Fig. 6. Example of a VT MÄK screen.

The model describes a subsystem with its equipment, into logical representations of its behaviour.

The models also have parameters that represent satellite conditions named satellite parameters such as temperatures, power, telemetry, orbital and operational modes (OM); Similarly, it has internal simulation parameters to configure the simulation, such as orbit number, visibility, simulated telecommands, interconnection parameters, etc. These models have rules that represent the logical association among the parameters [12].

The telemetry stores the value read by satellite sensors, i has a satellite parameter directly associated to. Other parameter is the OM, it represents a state of a given group of parameters within an established value range. An OM may represent the equipment, subsystem or the overall satellite condition.

A configuration is the set of initial values of the parameter groups at a given time.

Every parameter has a simulation step, which represents a sample or a change update frequency [13].

Simulation time is time considered to the simulation, it may be different from the current time, running faster or slower [13].

Modifications occurs via scripts (pre-configured conditions) or on-line (by a conductor), changing a parameter or parameter group. The models rules handle and propagate the changes.

A conductor is the simulation user that operates the simulator.

At the UIs, everything the conductor selects is an object, which is specialized into satellite, subsystem, equipment and parameter, etc.

The simulation architecture defines the structure of models, their interrelationship, and their principles and guidelines governing their design and evolution over time [13].

The simulator is a device, computer program or system that performs simulation. In training, it is a device which duplicates the essential features of a task situation and provides for direct human operation [13].

Simulation have many types and many categories [14], among them are the interactive simulations that have a human-in-the-loop (HITL), where the conductor controls the simulation/simulator acting in the interactive model.

Another name to HITL simulations is operational simulators [15], where states features that distinguishes them from simulators used for design. Monitoring and control of its models uses an incorporated user interface, so operators can receive training, validate procedures, investigate anomalous behaviour and respond to disasters before implementation.

III. GAMES

A game can be defined as a physical or mental competition in which the participants, called players, seek to achieve some goals within a given set of rules. In space research, a special kind of game is known: the war games. A war game is a simulation game in which participants seek to meet a specified military goal given pre-established resources and constraints; such as, a simulation in which participants make battlefield decisions and a computer determines the results of those decisions [13] .

Some simulation architectures came from attempts to solve war games, influencing the architectures of other simulations, standardizing it [11]. These architectures define the inner working of how the game works, and flows the models. In other hand, the game industry produced games; usually with low simulation fidelity, more emotional and aesthetic appeal.

The game industry has worked in storytelling and UI, using: simultaneous viewpoints, minimaps, contextualized information, 3D animation and information, natural gestures and data filtering. Just to name a few.

Simultaneous viewpoints offer views of the same evolving model from different angles and with different scenarios.

In games, the history is the evolving model that ties the scenes. Flight simulations generally represent simultaneous viewpoints in Head Up Displays (HUD), 3D views of pilot angle, outside of the airplane, from terrain sites and from all map positions. Figure 7 illustrates these possible different views being (a), (c) and (d) from the Microsoft Flight Simulator[5] and (b) from Google Earth Flight Simulator[6].

Maps and mini-maps provide a macro view of the system, showing all current situation. It does not give specific information, but tracks global changes [8]. Figure 8 shows two examples (a)[7] and (b)[8] with mini-maps, that usually are on a screen corner; and (c) shows a map placed on the full screen.

Contextualized information is a menu representation of data dedicated to the selected object. Games often use this type of menus, to give possible actions based on the object context.

3D animation and information provide the problem abstraction that reflects the internal simulation elements. It is important to track the simulation model and map the correct animation.A way to integrate a computer simulation with computer graphics is described in [16], which he tackles as three reasonable issues: software architecture, software coordination and indeterminate mapping:

Fig. 7. Multiple Views examples.

Fig. 8. Mini-maps and maps examples.

- Software architecture: defines the structural elements, their relationship and the way that they bind.
- Software coordination: defines a transparent communication management among elements.
- Indeterminate mapping: defines an abstraction virtual content scene that might represent the simulated data.

Natural gestures captured by, current cheaper, hardware provide a more human-oriented commands, enriching the problem metaphor. Kinect, and PrimeSense[9] like hardwares are common for 3D gesture recognition. Similarly, Smartboards[10], Wiimote whiteboards[11], tablets are common to 2D gestures.

Data abstraction and filtering helps to contextualize the problem and understand it. A bigger picture provided by the maps is explored by filtered contexts, at each 3D view a determined set of elements. This allows a top-down understanding, and exploration.

Game interfaces (excluding MMORPG[12]) usually have clean interfaces with minimum information. Actions are contextualized to location and objects, without strong hierarchical menus to find the correct attitude. Some operational system

[5]http://www.microsoft.com/games/fsinsider/
[6]https://support.google.com/earth/topic/23746?hl=en
[7]http://www.needforspeed.com/
[8]http://us.blizzard.com/pt-br/games/wow/
[9]http://www.primesense.com/
[10]http://smarttech.com/smartboard
[11]http://johnnylee.net/projects/wii/
[12]Massively Multiplayer Online Role-Playing Game

are incorporating this behaviour and providing a standard but configurable way, as the Windows 8 charm bar[13], shown in Figure 9, where the same bar, shows contextualized information of the selected software.

Fig. 9. Windows 8 context charm bar.

These and others game interaction technologies are migrating to every-day interfaces, and to operational simulators. Next session shows a migration process from a second generation to a third generation UI, based on traditional components to game like components transformation.

IV. Interface Transition

Transformation of one UI to another demands a step by step process for understanding the components and the changes to improve the desired usability.

A. The current Interface

The current interface intends to show specialists the subsystems behaviour, so they can evaluate interrelation effects of normal or abnormal behaviour. They have to know parameters acronyms and their relation to validate graphs, values and ranges.

This interface allows overview of tables, graphs, exploring and inspecting specific values, keep the simulation time change and its configurations. The interface is QT[14] based allowing fast graphical user interface (GUI) creation. Figure 10 shows the current interface elements.

In this type of interface it is hard to keep track of the main changes, it diminishes the behaviour abstraction in favor of parameter analysis. The specialist knows what and where to look, which is more a bottom-up understanding, the specialist looks into the parameters to understand them all.

[13]http://windows.microsoft.com/en-us/windows-8/meet
[14]http://qt-project.org/

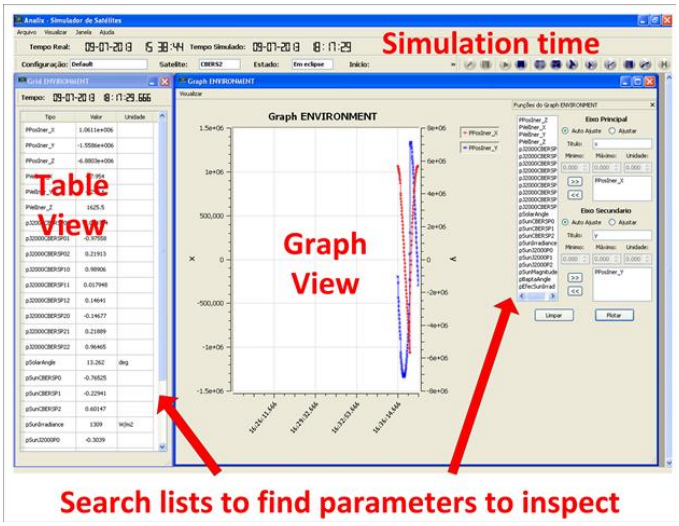

Fig. 10. Current Satellite Simulator UI.

B. The proposed interface

This interface intends to use game-like visual components to offer larger picture (maps and minimaps), with filtered contexts. It attempts to allows specialists and non-specialist to search for information and seek the desired level, hence a top-down approach.

3D representations (model + animation) illustrate operational modes, representing if it is operating in normal, stand-by, off or emergency condition. This allows faster understanding of what is happening throughout the system.

This proposed interface decouples the UI from the simulator, differently from earlier versions. The UI-Simulator connection occurs via a simulation architecture run-time interface. Therefore the simulation core is auto-sustained and works independently, so that the UI follows its the updates and triggers changes.

The implementation toolset to this new interface tries to answer four questions: How to see and filter data? How to inspect and search parameters? How to follow changes? How to interact? Some details are highlighted below:

1) Implementation framework: This UI uses an open source framework, the openFrameworks (oF)[15], to handle 3D models, XML files, fast calculations, portability through operational systems and open to further add-ons. oF shows as a more appropriated choice, among some possible frameworks as Java (Processing) or Action Script (Flash) counterparts [17], providing highly reusable components, expandability characteristics, and interoperability in C++ language.

2) How to see and filter data?: The earlier interface sees and filters data by menus, displaying data by Table View, with its current values; or by evolution inspection into a Graph View.

Data are facts and statistics collected together for reference or analysis [18]. In this simulation the parameter type that groups other parameters is the OM. Hence it is necessary

[15]http://www.openframeworks.cc/

to reproduce the OM status of each major object (satellite, subsystem and equipment) which 3D visualization can address. Abstract behaviour maps can contextualize pictures, refining data to inspect the satellite OM, the subsystems OM and the equipment OM. These views are called, respectively, as Space View, Satellite View, and Subsystem View.

Building a game avatar with several modes and reactions, requires elaborating a way to transit among the possible states. The solution is a state machine, or a finite state machine (FSM), it is a mathematical model used to represent a sequential logic events, the machine is on only one state at a time, and it changes states triggered by events or conditions, called transitions [19]. A FSM can be directly implemented using the State Design Patterns [20].

An example of FSM is show in Figure 11, where in its Space View it is possible to select a satellite and to request to filter data, changing to Satellite View. The Satellite View presents the internal structure in means of subsystems, and allows filtering data again, changing to Subsystem View. The Subsystem view presents only the pieces of equipment that compose the subsystem. Further refinement is possible, but it was not present at the time of this paper.

oF allows a simple way to interact with 3D models, called ofEasyCam[16]. It is a simple implementation with an already built mouse interaction, allowing rotation, translation and scale from the origin axis; focusing in other object allows changing of the axis position, such as, to track the movement by Earth or satellite point of view.

The View's control is a state machine where Views are instances of an abstract element View. The active View represents the active state that process the interaction. An example of Views interrelation is shown in Figure 11, with the four 3D Views showing the space elements (Space View), the satellite subsystems (Sat View), the subsystem equipment (SubSystem View) and the inner parameters (Equipment View); a timeline view keeps past and future events sequence (Timeline View); a system menu view configures and changes simulation settings (System View); a parameter view shows the simulation object hierarchy (Param View); and a 2D/3D graph (Graph View). By selecting and dragging events swap between Views.

3) How to inspect and search parameters? : The earlier interface requests the conductor to inform what subset of parameter he/she desires by a list, to start the table or the graph.

Touch or Kinect based does not have the precise pointing of a mouse, so it is necessary larger structures to allow better pointing. The 3D Model Views already allows navigating between contexts to search a parameter through the View level. This deals with the following problem, the specialist conductor knows the parameter that he desires and does not want to navigate to find it. How to search through a list of parameters, and pick one? Without a menu? The answer came from Reconfigurable Disk Tree (RDT). RDTs are a new visualization technique that alleviates the searching process of large hierarchies while maintaining its context. Each node has a disc around which its children parameters. [21]

[16]http://www.openframeworks.cc/documentation/3d/ofEasyCam.html

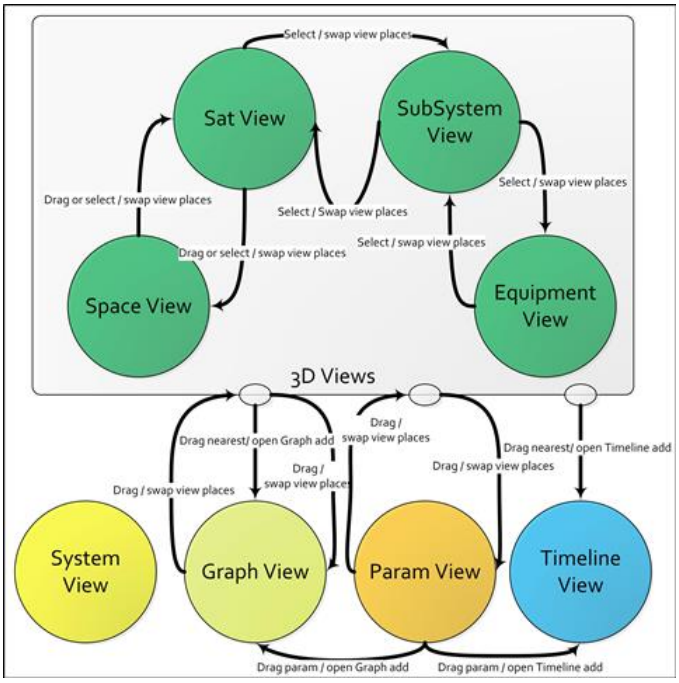

Fig. 11. Relationship of the various Satellite Simulator views.

The RDTs need a four step algorithm to create and show the diagram:

1) Load: Store the hierarchic tree into polymorphic nodes (satellite, subsystem, equipment, parameters) from an abstract node (NodeRDT);
2) Calculate Radius: Recursively calculate the size of the disks and define the radius opening of each inner father disk node;
3) Calculate Positions: To draw the 3D representation by the Cartesian OpenGL commands, and to change the focus of the tree is necessary to know each x,y,z coordinate, calculating it recursively;
4) Draw: As all the nodes coordinates are already calculated by earlier steps, the drawing is a transition through nodes, placing the information.

A RDT of the simulated system is shown in Figure 12 with the proportion of distributed elements, from Earth to available equipment.

In order to inspect a parameter a "point and click" action, from a touch interface or the mouse opens a contextualized panel specialized on the selected object at a selected place. So the information is View and Object sensitive. An oF add-on called ofxUI[17] allows a clean contextualized information UI to create menus, on the fly, gathering and embedding GUI blocks of traditional, but in different shape, buttons, drop down menus, radios. Figure 13 shows the possibilities example of the ofxUI.

4) How to follow changes?: The earlier interface allows to follow parameters evolution by graphs or by the conductor perception of the instantaneous value change of the tables. As

[17]https://github.com/rezaali/ofxUI

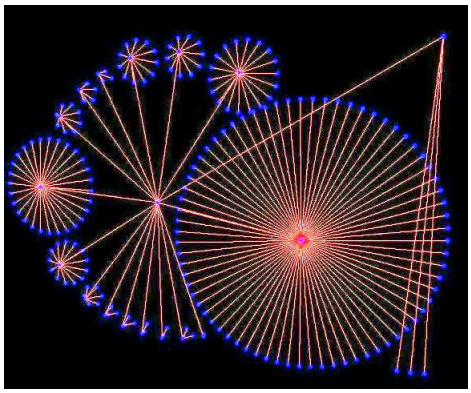

Fig. 12. Reconfigurable Disk Tree with the simulated system elements.

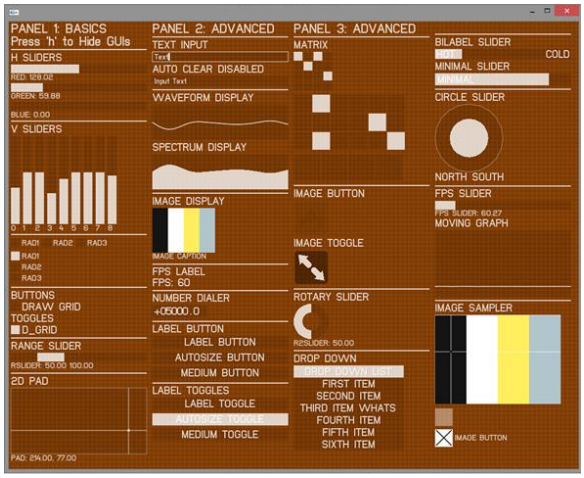

Fig. 13. An ofxUI example that allows a clean contextualized information panels.

a bottom-up strategy the conductor selects what he wants to see and follows it.

At the 3D Model Views, it is possible to animate the objects to represent the OM, tracking the major parameter changes. Inspecting each parameter and looking into their contextualized panel also allows an instantaneous value tracking and even a simple time relation change, provided by ofxUI component called moving graph. However these solutions provide high abstract information or single parameter tracking, not a group or interrelation is possible, only if already set as a possible parameter from the simulation. A better solution to address single parameter tracking still lies on graphs.

5) How to interact?: The interaction into 3D interfaces by a 2D screen, to allow touch or mouse interaction, requires constantly projection to the screen. Therefore, every 3D View has to project on 2D, the ofxEasyCam oF add-on allows projection calculation to convert between spaces, as: worldToScreen(), screenToWorld(), worldToCamera() and cameraToWorld().

In this case to bring the world to screen, a worldToScreen function, as in the recursive code at Figure 14.

To control the user input, on oF, the interaction was generalized into the following steps:

1) Continuously track of user position and closer interaction enabled objects at onUserMove;
2) At an onPress event get the start position of the movement;
3) At an onRelease event get the final position and set a drag or click/hold action, passing it to be treated by the respectively View;
4) The View decides if it throws a panel, moves the camera, drags something or passes the decision to the nearest object action handler.

```
void findNearestNode(NodeRDT* start, userAction *ua){
        if(start!=NULL){
        ofVec3f x = this->cam.worldToScreen(start->place);
        float distance = x.distance(ua->actual.position);
        if( (this->nearestIndex == 0 ||
                start->distMouse < this->nearestDistance) ) {
           this->nearestDistance = distance;
           this->nearestNode = start;}
        nearestIndex++;
        vector<NodeRDT*>::iterator itNode;
        for(itNode = start->nodes.begin();
                itNode != start->nodes.end(); itNode++)
           findNearestNode((*intoed),ua);
        }
}
```

Fig. 14. Adjusting coordinates using a wortToScreen recursive function.

3D Views allow global interactions and panel the specific interactions. However, sometimes, the positioning of the model on the 3D View is not easy by a mouse/touchscreen interaction. A solving attempt may use an Augmented Reality(AR) [22] tracking, so the user adjust the 3D Model and visualize at better angles. The oF has an ofxARToolKitPlus add-on [17] that handles the AR. It allows capturing the marker's transformation matrix to View's camera adaptation. Further development intends to allow this interaction by Augmented and Cross-Reality [23] interaction .

The Space View point the 3D orbital representation around the Earth Globe with its Earth Stations is shown in Figure 15 with the simulation time and the auxiliary Views positions;

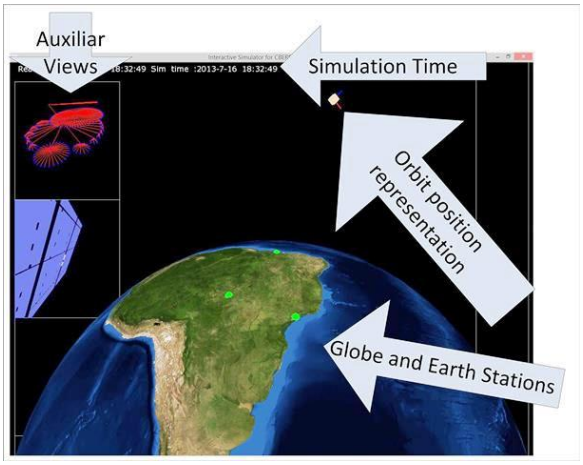

Fig. 15. Space View with Earth, satellite, earth stations, simulation time and auxiliary Views.

The Space View with a change on the focus to the satellite is shown in Figure 16. In order to look into its point of view, it can be used to simulate the cameras visible range, and keep

attitude maneuver.

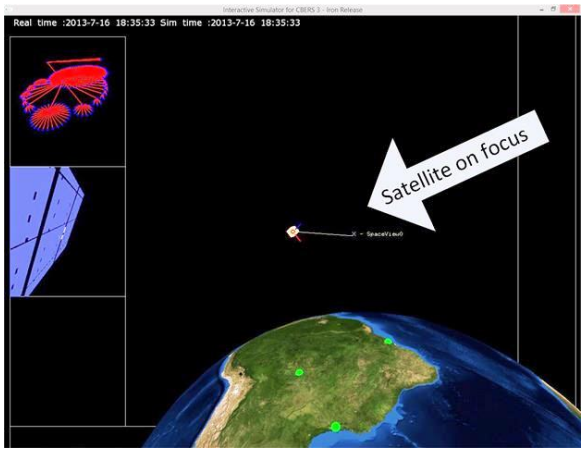

Fig. 16. Space View with satellite focus.

The Satellite View, with a Subsystem element example, the Solar Array Generator is shown in Figure 17. A red point with a line shows the nearest object selection; a clean Context Panel and a mini-map to keep the orbital context active.

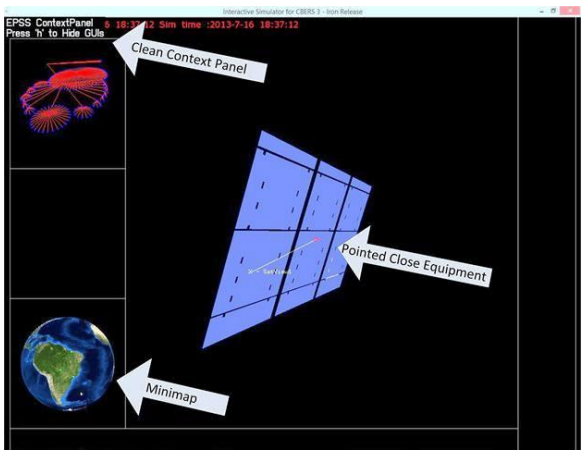

Fig. 17. Satellite View example with nearest object selected.

Finally, the Param View with the open RDT with all elements is shown in Figure18. Note that in the RDT it is possible to see the amount of equipment of each subsystem. Selecting a node, by the nearest nodes allows to open its Context Panel.

V. Evaluation

In order to evaluate the interface, the users were asked to perform the following tasks:

- Find and change a parameter;
- Find an equipment;
- Identify which object is on fault condition;

Twenty participants answered a questionnaire (The evaluation results are illustrated in Figure 19), created in the Google Drive Form. The questions inquired issues about interaction,

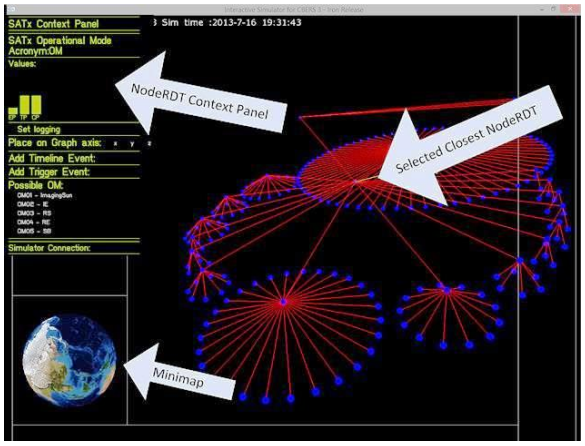

Fig. 18. Satellite Parametric View (Param View).

interface, visual aspects, motivation, behaviour and game access [24]:

- Interaction: (1) How easy was to find an equipment?(2) How suitable was the visual cues? (3) How was the response time of the interactions?
- Interface: (4) Was the information placement adequate? (5) Is the interface attractive? (6) Is it fast to learn?
- Visual Aspects: (7) Are the colors in harmony? (8) The texts are readable? (9) How are the quality of 3D objects?
- Motivation: (10) How hard is to keep attention? (11) Is the content association correct? (12) The degree of satisfaction?
- Behaviour: (13) How are the feedbacks of near objects? (14) How is it adapted to different stakeholders?
- Game access: (15) Is it simple to launch? (16) Is it easy to access the views? (17) Is it easy to search and inspect parameters?

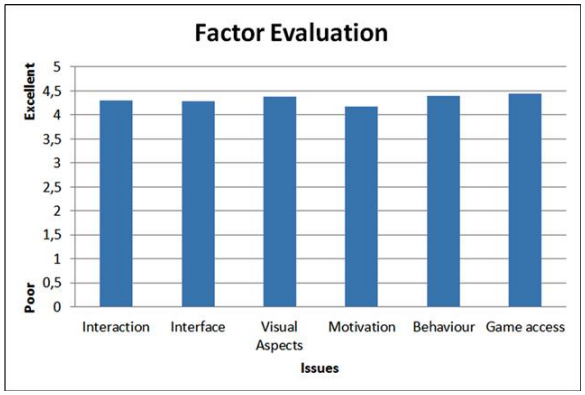

Fig. 19. Survey results concerning user's satisfaction.

Some comments involving other possible interaction approaches, as Kinect based. The majority of answers says that they did not desired it; saying that the mouse interaction is more typical. The AR possibility to change the view place

was seemed as nice, but not easy, they suggested it to be more as presentation tool. This shows that this stakeholders group prefer UI 2D handling, even if UI objects are in 3D.

VI. Conclusion

In this work a UI attempt was introduced for a new interaction approach, from a bottom-up to a top-down discovery, allowing non-specific engineers to understand the overall system picture as much as the specific engineers and decide which sub context to explore. This interface allows a broader types of conductors.

The type of interface employed does not supersede the earlier UI as it is faster to specific engineers, which have exact knowledge of the desired parameters.

The design of a top-down interface requires a search of possible manipulations and visualizations methods. Nowadays computing power allows game-like interaction techniques, which enhances the conductor experience.

The new generation of stakeholders, who have experienced games, seek new computer UIs and untraditional human-machine interactivity as those of game-like interfaces. Some sort of real science fiction applications, a reality now on the daily basis activities, with augmented reality guides to 3D interactive are possible in operational satellite simulators.

Acknowledgment

The authors would like to acknowledge the help provided by Dr. Claudio Kirner, which have constructively added suggestions during this research work, the CNPq for the scholarship provided, Dr. Antonio Carlos de O. Pereira Jr. and Dr. Nilson Sant Anna for the case-study support and Lincoln Azevedo for the 3D models provision

References

[1] ESA, "Ecss-e-tm-10-21a - space engineering - system modelling and simulation," ESA, Noordwijk, The Netherlands, Tech. Rep., 2010. [Online]. Available: http://www.ecss.nl/

[2] "National institute for space research website," INPE, 2013. [Online]. Available: http://www.inpe.br

[3] ESA, "Ecss-e-st-70c - space engineering - ground systems and operations," ESA, Noordwijk, The Netherlands, Tech. Rep., 2008. [Online]. Available: http://www.ecss.nl/

[4] M. L. d. Oliveira e Souza and G. d. C. Trivelato, "Simulators and simulations: their characteristics and applications to the simulation and control of aerospace vehicles," in *SAE Technical Paper*, International Congress and Exposition of the Mobility Technology, 12. (SAE). São Paulo: SAEBrasil, 2003, p. 9. [Online]. Available: http://papers.sae.org/2003-01-3737

[5] L. C. Botega and P. E. Cruvinel, "Realidade virtual: Histórico, conceitos e dispositivos." in *Aplicações de Realidade Virtual e Aumentada*, SBC, Ed. Porto Alegre: SBC, 2009.

[6] B. A. Myers, "A brief history of human-computer interaction technology," *interactions*, vol. 5, no. 2, pp. 44–54, Mar. 1998. [Online]. Available: http://doi.acm.org/10.1145/274430.274436

[7] J.-M. Oh, Y. S. Lee, and N. Moon, "Towards cultural user interface generator principles," in *Multimedia and Ubiquitous Engineering (MUE), 2011 5th FTRA International Conference on*, 2011, pp. 143–148.

[8] J. Preece, Y. Rogers, and H. Sharp, *Beyond Interaction Design: Beyond Human-Computer Interaction*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[9] S. Wenzel, J. Bernhard, and U. Jessen, "A Taxonomy of visualization techniques for simulation in production and logistics," in *the 2003 Winter Simulation Conference*, S. Chick, P. J. Sanchez, D. Ferrin, and D. J. Morrice, Eds., 2003, pp. 729–736.

[10] C. Abt, *Serious games*. Viking Press, 1970. [Online]. Available: http://books.google.com.br/books?id=5z-QAAAAIAAJ

[11] J. Dahmann, R. Fujimoto, and R. Weatherly, "The dod high level architecture: an update," in *Simulation Conference Proceedings, 1998. Winter*, vol. 1, 1998, pp. 797–804 vol.1.

[12] J. Tominaga, C. S. Cerqueira, J. Kono, and A. M. Ambrosio, "Specifying satellite behavior for an operational simulator," in *Proceedings of SESP2012*, Simulation and EGSE facilities for Space Programmes (SESP 2012). SESP 2012, 2012. [Online]. Available: http://congrexprojects.com/docs/12c09_docs/s7_1400_ambrosio.pdf?sfvrsn=2

[13] DoD, "Dod modeling and simulation glossary," DoD, Tech. Rep., 2010. [Online]. Available: http://www.msco.mil/files/Draft_MS_Glossary_March_B_version.pdf

[14] W. T. Lord, "Air force modeling and simulation vision for the 21st century," Headquarters United States Air Force, Tech. Rep., 2010. [Online]. Available: http://www.afams.af.mil/shared/media/document/AFD-100930-038.pdf

[15] C. Morris and D. Rothwell, "Operational spacecraft simulations: present and future," *Simulation and Modelling of Satellite Systems*, pp. 9–9(0), 2002. [Online]. Available: http://digital-library.theiet.org/content/conferences/10.1049/ic_20020069

[16] G. S. Lee, "Towards an integration of computer simulation with computer graphics," in *Proceedings of the Western Computer Graphics Symposium*, 1999.

[17] C. S. Cerqueira and C. Kirner, "Introdução à utilização de openframeworks para o desenvolvimento de aplicações de rva," in *Tendências e Técnicas em Realidade Virtual e Aumentada*, A. Cardoso, E. Zorzal, M. de Paiva Guimarães, and M. Pinho, Eds., no. 3, SBC. Porto Alegre: SBC, May 2013, pp. 250–278.

[18] "Meaning of data," Dictionary.com, 2013. [Online]. Available: http://dictionary.reference.com/browse/data

[19] I. Millington and J. Funge, *Artificial Intelligence for Games, Second Edition*, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009.

[20] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns*. Reading, MA: Addison Wesley, 1995.

[21] C.-S. Jeong and A. Pang, "Reconfigurable disc trees for visualizing large hierarchical information space," in *Proceedings of the 1998 IEEE Symposium on Information Visualization*, ser. INFOVIS '98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 19–25. [Online]. Available: http://dl.acm.org/citation.cfm?id=647341.760271

[22] R. T. Azuma, "A survey of augmented reality," *Presence: Teleoperators and Virtual Environments*, vol. 6, no. 4, pp. 355–385, Aug. 1997. [Online]. Available: http://www.cs.unc.edu/~azuma/ARpresence.pdf

[23] C. Kirner, C. S. Cerqueira, and T. G. Kirner, "Using augmented reality cognitive artifacts in education and virtual rehabilitation," in *Virtual Reality in Psychological, Medical and Pedagogical Applications*. InTech, 2012, pp. 247–270.

[24] C. Kirner and T. Kirner, "Development of online educational games with augmented reality," in *EDULEARN13 Proceedings*, ser. 5th International Conference on Education and New Learning Technologies. IATED, 1-3 July, 2013 2013, pp. 1950–1959.