VOL. 9, NO. 12, DECEMBER 2014 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

COCKPIT VIEW VEHICLE GAME SIMULATORS FOR MASSIVE PARALLEL ARRAYS PLATFORM BASED ON NEURAL-FUZZY SYSTEMS

Felix Pasila[1], Gregorius S. Budhi[2], Andreas Sutantra[2] and Hans Juwiantho[2]
[1]Electrical Engineering Department, Petra Christian University, Surabaya, Indonesia
[2]Informatics Department, Petra Christian University, Surabaya, Indonesia
E-Mail: felix@petra.ac.id

ABSTRACT

This research is an extended investigation of using game simulator to the Massive Actuator Array platform. Two kind of prototypes cockpit view game simulators are designed to be implemented to the platform. The two prototypes are Aircraft and ground All-terrain vehicle game simulator. The aircraft simulator is made by OGRE3D engine and SDL, while the ATV simulator used NVidia PhysX engine. The controller of aircraft simulator is a joystick and/ or keyboard buttons and the controller of ATV simulator are steering wheel and/ or keyboard. The output of this neuro-fuzzy network will be used to control the movement of the actuator Massive Array platform. From the simulation results we concluded that the two game simulators prototypes have been made correctly. In addition, some constraints are also added to the simulator in order to give real environment, such as clouds and wind effects on aircraft as well as the effects of flat road, off road, ice, sand and mud on land vehicles. Regarding the Simulator performances show that both prototypes are suitable to be implemented to the Actuator Massive Array platform.

Keywords: MA^{3}I, vehicle simulator, computer game, artificial neural network.

INTRODUCTION

This research is a continuation of a research name: MA^{3}I: Massively Actuator Array using Artificial Intelligence. This kind of manipulator is a very special kind of mechanisms whose actuators can only be made switching between two or three possible states. Moreover, the designed platform has an effort to consider sensor-less manipulators as well as to reduce the computing time and the complexity of computer interfacing, as described in [1]. The proposed research have two kind of prototypes software cockpit view simulator which are designed to be implemented to the Actuator Massive Array platform (Figure-1).

Figure-1. Binary Platform MA^{3}I [1].

For instance, there are two kinds of softwares cockpit view simulator games that propose here: Airplane Simulator and ATV Simulator. Both simulators can be described as follows:

Airplane Cockpit View Simulator

A Prototype of Aircraft Cockpit View Game Simulator which operates in a virtual world space. The controller of this kind simulator is a joystick and/ or keyboard buttons. The simulator software will have the flexibility of the movement in three axis of rotation, name: Pitch, Yaw and Roll (see Figure-2). The effects of aircraft motion will be vizualized on the monitor screen. Another output is a data of aircraft movement to train an artificial neural network algorithm Neuro-Fuzzy Takagi Sugeno type MIMO (NFTs) or other types. The output of this neural network will be used to control the movement of the actuator Massive Array platform.

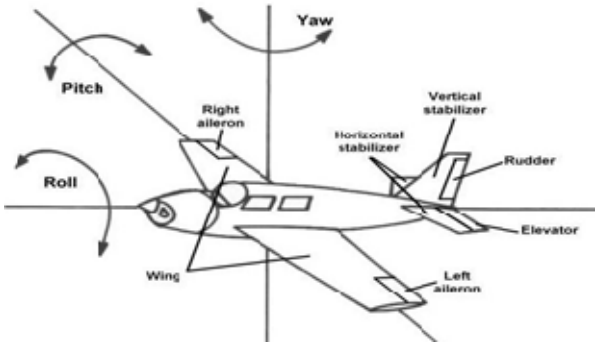

Figure-2. Airplane Movements.

Land All-Terrain Vehicle Cockpit View Simulator

A prototype of ground All-Terrain Vehicle (ATV) Cockpit View Game Simulator running on a simple virtual world in the form of inroad terrain such as normal, snowy, sandy, and muddy roads. And also the rugged off-road terrain, mountains, up and down. Controls of this simulator used steering wheel and/ or keyboard. This car can turn left and right and can move forward and backward. The effects of the movement of the car and also the effects that arise due to the form of terrain shape that is

crossed will be visualized on the monitor screen. Another output is the training data from ATV car movement that will be trained on an artificial neural network algorithm Neuro-Fuzzy Takagi Sugeno type MIMO (NFTs) or other types. The output of the neural network will be used to control the movement of the actuator Massive Array platform.

Motion Simulator

Motion Simulator is a hardware unit that can move along and be controlled by vehicle simulation program that is played in it. The purpose of this hardware is to create a sense of verisimilitude: Where players are looking for a really feel nuances of driving, flying, or vehicle control. The main goal of our research is to build a motion simulator by way combining the cockpit view simulators that we built with the actuator massive array platform. The simulators will control the movement of the actuator massive array platform using a mechanism that we will build later using artificial neural network concept. On Figure-3 can be seen the block diagram of motion simulator design.

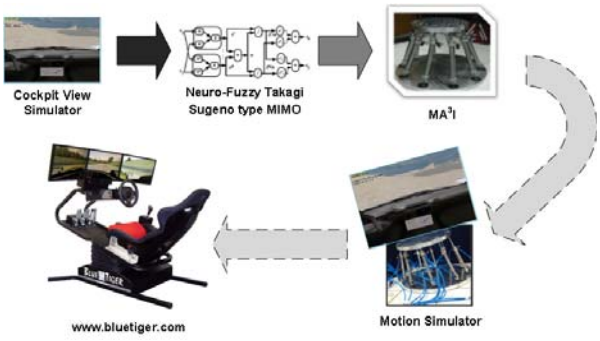

Figure-3. Motion Simulator Design Block Diagram.

GAME AND VIDEO GAME DEFINITION

A game is an activity that is done in the context of pseudo reality (pretended reality), where the participants try to reach at least one purpose of the game which is not easy (nontrivial) in the way of action that follow the rules in the game [2].

However, the video game is a game played by the electronically manipulating images produced by a computer program on a monitor or other display [3].

GAME GENRE: SIMULATION

The game simulation (Sims) will make the gamers experienced the real-world situations which is safe, practical and controlled. Since 1950, the Department of Defense has trained military with computerized simulators such as flight Sims, tank Sims, and war gaming Sims [4].

The simulation should be interesting. Where the actors can feel the real experience when they use the application. Most computer simulators attempting to simulate real-world conditions which very dangerous or very expensive. Exploring and reproducing training scenarios will be more practical and easier when set in the simulator. Simulation applications can be classified into two parts, name: vehicle simulation and management simulation. Vehicles simulation such as trucks, airplanes, helicopters, ships, submarines, spacecraft, space stations, motorcycles, and so forth. While management simulation including managing nuclear power plants, predicting the stock market, became the mayor or president of the United States, became the owner of a golf club, becoming manager of the city zoo, the Roman Emperor, became the owner of the amusement park (rollercoasters and rides), and even managing the life of an ant colony [4].

A simulation is a simplified model of reality and judged by its realism and correspondence to the system which it represents. A game is created without any reference to reality, what is never the case for a simulation or a game simulation. It is not necessarily present a conflict or a competition in simulation. Person who uses the simulation is not looking to win, what is the case in a game [5].

Vehicle Simulation

Vehicle simulation includes several kinds of environment, and game mechanics. They can be in the air, on land, in water, or in space. They can include race against another player or artificial intelligence, or they may involve exploration or simply the experience of using the vehicle. The most common element is a sense of verisimilitude: Where players are looking for the true experience - really feel nuances of driving, flying, or vehicle control. Since most of us have a car so we know how it feels. But most of us do not know what it's like driving a car at speeds of 200 miles per hour or driving a car which have installed and used a variety of weapons to fight. Most of us have not experienced fly the plane ourselves, but we know what was on the plane. We certainly have not been felt and imagined fly a fighter plane too. Simulation varies from realism that represent how to drive a vehicle on the road or in the air, up to adding specific mechanisms such as fighting, racing, and other special challenges such as skid or slaloms, and so on [2]. Picture of vehicle simulation can be seen in Figure-4 and Figure-5.

Figure-4. Airplane Cockpit View Simulator.

Figure-5. Land Vehicle Cockpit View Simulator

OBJECT-ORIENTED GRAPHICS RENDERING ENGINE (OGRE)

OGRE3D is a mature, reliable, flexible, cross-platform, and full-featured library for use in developing real-time 3D graphics. OGRE is design-led rather than feature-led. Every feature that goes into OGRE is considered thoroughly and slotted into the overall design as elegantly as possible and fully documented [6]. OGRE is designed to provide not only a world-class graphics solution but also for other features such as sound, networking, AI, collision, physics, and etc., you can integrate OGRE with other libraries. OGRE is under the MIT license and can use it for free as long as it includes the contents of the "COPYING" file somewhere on the application [7]. Ogre3D engine is used in the manufacture of prototype aircraft cockpit view simulator.

SIMPLE DIRECTMEDIA LAYER (SDL)

Simple Direct-media Layer is a cross-platform library designed to provide low level access (low-level access) to audio, keyboard, mouse, joystick, and OpenGL and Direct3D graphics hardware. It is used by the video playback software, emulators, and popular games. SDL officially supported Windows, Mac OS X, Linux, iOS, Android and several other platforms [8].

This research will be using the library of SDL joystick to control an aircraft simulator cockpit view.

PHYSICS FOR AIRPLANE SIMULATOR

There are four major forces that act on an airplane in flight: gravity, lift, thrust, and drag (See in Figure-6). Gravity is the force that pulls the aircraft to the ground. Lift is the force generated by the wings of the aircraft to counteract gravity and enable it to stay aloft. Thrust force generated by the aircraft's jet engine or propeller, it increases the aircraft's velocity and enables the lifting surfaces to generate lift. The last one is Drag, a force that counteracts the thrust force, impeding the aircraft's motion [9].

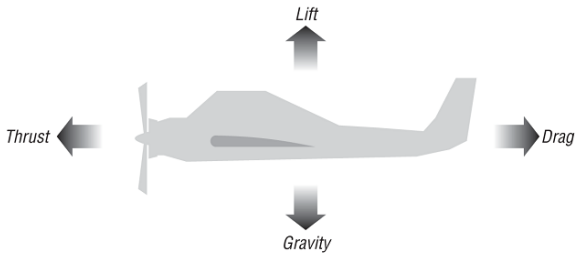

Figure-6. Forces on aircraft in flight.

We don't need to further explain the thrush because it directly comes from simulator pedal (user input). The gravity is defined by = 9.81 m/s^2 [10].

Lift and Drag

The formulas to estimate the magnitudes of lift and drag forces on the wing section are [9]:

$$L = C_L (1/2) \rho V^2 S \quad (1)$$

$$D = C_D (1/2) \rho V^2 S \quad (2)$$

where:

L and D	= Lift and Drag
C_L and C_D	= Lift and Drag Coefficients
ρ	= Air density
V	= Speed through the air (Velocity)
S	= wing platform area (span times chord for rectangular wings)

This very simplified approach does not account for span-wise flow effects, the flow effects between adjacent wing sections and air disturbances such as downwash.

NVIDIA PHYSX

NVidia PhysX SDK is a physics engine that is used in a game. PhysX is an important role for most game makers. This is because the game makers no longer need to program their own physics formulas you want to use in the game, considering the very complex physics interactions in the modern game. Features that are present in PhysX SDK include continuous collision detection, ray casting and shape sweeps, solvers for rigid body dynamics, fluids, particles, vehicle and character controllers [11].

The version used in this study is PhysX-3.3.0. This version can perform rendering using Microsoft DirectX. PhysX SDK is used in the manufacture of prototype ground vehicle ATV cockpit view simulator. With PhysX, we don't need to implement physics formulas by our own.

MIMO-TS TYPE NEURO-FUZZY NETWORK

The structure of MIMO-TS type Neuro-Fuzzy Network we use here is the same with that one described in [12, 13] (See in Figure-7).

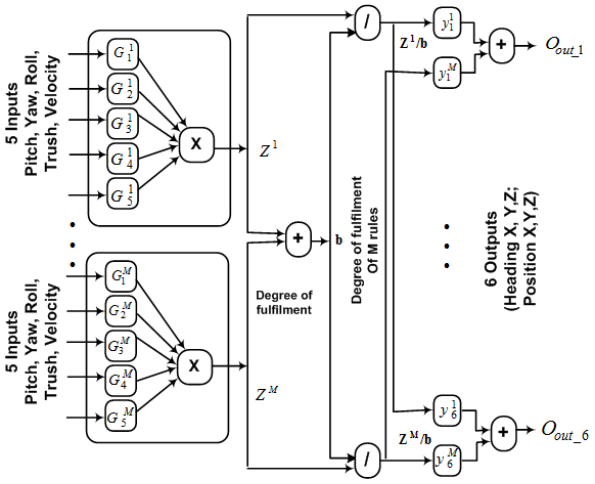

Figure-7. MIMO-TS type Neuro-Fuzzy Networks representation of Airplane Simulator Model as modified from [12, 13].

Neuro-Fuzzy Networks (NF) is already well-known as one of universal approximation's tools in computational intelligence field that combines the learning ability from neural network and rule-based interpretation from fuzzy systems. Because of this reason, an NF is widely used to determine optimal parameters from the learning mechanism efficiently.

APPLICATION DESIGN

In general, the gameplay of both prototype simulator games is made so easy that is: While playing the player must operate the vehicle through several checkpoints have been made from start to finish. This simplification is done because the focus is to design simulation that can be implemented well to actuator Massive Array platform, and also how these two applications can produce output data that can be used for training of the neural network controller platform movement.

Furthermore, the design of the second prototype made can be seen in the block diagram in Figure-8.

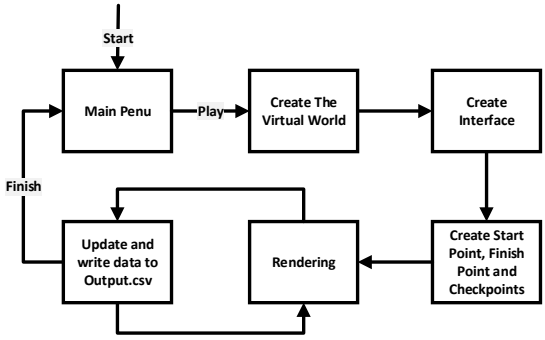

Figure-8. Block Diagram Simulators.

When running, the application will be heading to the Main Menu. Furthermore, when the player chooses Play, the application will make the Virtual World application then make the Interface. Then the application will generate the Start Point, Finish Point and also some Checkpoints. Furthermore, all initial conditions will be rendered by the application and then the application waiting for input from the player. When a player has entered an input through input devices, the application will calculate the position of the next Vehicle, write position data and other data in the file Output.csv. Then the application will re-rendering the new position of vehicle, and then waits for player input again. So forth rendering and updating processes is done repeatedly until player vehicle position reach the finish line. When finish position is reached then the application will return to the Main Menu.

Virtual World made with the help of the height-map set/ drawn on the image processing applications such as Photoshop and others, and then saved to a specific format according to the engine used. Height-map example can be seen in Figure-9. Height map will determine the height of the terrain, the smaller the grayscale values in the image (light), the higher the value of the Y axis for the location and vice versa. Furthermore, the file is invoked on the application program that created the simulator vehicle. Similarly to the particle, vehicle and other objects of the data set on the folders that correspond to the format of the engine is used.

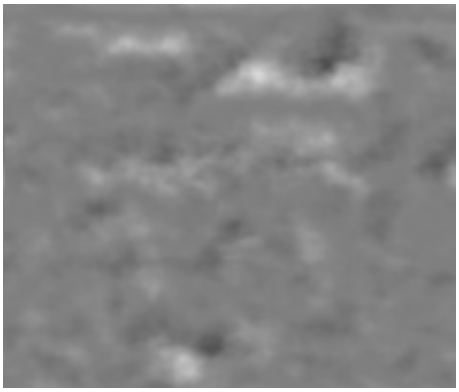

Figure-9. Height-map Example.

The Interfaces of the aircraft and ground ATV simulators can be seen in Figure-10 and Figure-11.

Checkpoints

In each prototype simulator, there are several checkpoints as the goal of the player from the start to finish, if one checkpoint is missed then Finish points will not show up. Checkpoint of an aircraft simulator is a wireframe cube shape as shown in Figure-10. As for the ATV checkpoint shaped like colorful lines as shown in Figure-11.

Figure-10. Interface and Checkpoint of Airplane Simulator.

The checkpoint data is saved in a file destination.txt. But between aircraft simulator and ATV simulator they have a little bit different format. For ATV simulator just need one coordinate for each data (X, Y, Z), while for aircraft simulator the size of data cubes added to the axis X, Y and Z (length, width and height).

Figure-11. Interface and Checkpoint for ATV simulator.

Cloud and Wind Design for Airplane Cockpit View Simulator

Making clouds in Airplane Cockpit View Simulator is not easy considering the engine used does not provide functionality to generate clouds. Therefor the cloud is made from many spheres that are generated randomly in a certain position of a virtual cube. The position of a virtual cube and other data are stored in a txt file with format as shown in Table-1.

Table-1. Format of Cloud.txt.

Name	Definition
Cube X1 Coordinate	Define start point of virtual cube
Cube Y1 Coordinate	
Cube Z1 Coordinate	
Cube X2 Coordinate	Define end point of virtual cube
Cube Y2 Coordinate	
Cube Z2 Coordinate	
Spheres Counts	Defines the number of spheres that are used to form clouds
X-axis Range	Defines the maximum magnitude of each cube which randomly generated to form clouds
Y-axis Range	
Z-axis Range	
Red	Define the cloud color in RGB format
Green	
Blue	

Furthermore, when the aircraft enters a particular cloud position will be simulated by the common vibration movements. It is done by rotating a bit up and down at random direction toward the plane on the way to the next position. Examples of cloud and turbulence effects can be seen in Figure-12.

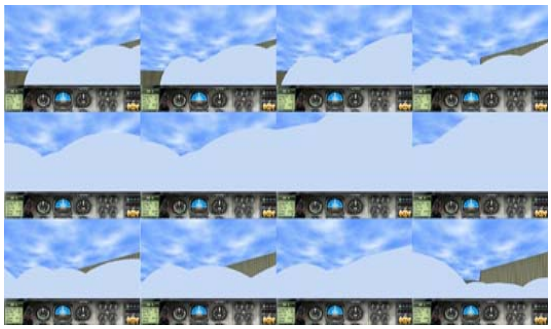

Figure-12. Example of Cloud and Shaking Effects.

To simulate the wind, will be randomly generated multiple virtual cube positions for the wind to occur, wind direction and wind speed. When an aircraft enters a virtual cube position to the wind, the aircraft position will be slightly translated/ driven in the direction and the speed of the wind.

Output File

File formatted as *.csv is created by recording every movement of the position of the vehicle driven by the player. The usefulness of this file is to be used as training data for a neural network-type MIMO Neuro-Fuzzy type Takagi Sugeno have been trained will be used to control the movement of the actuator Massive Array platform so that the movement of the platform will be

aligned with the vehicle motion in cockpit view simulator. The design of Output.csv file contents for Airplane Simulator can be seen in Table-2, while it is for the ATV simulator can be seen in Table-3.

Table-2. Format of output.csv for Airplane Simulator.

Name	Definition
Dx	Pitch Value
Dy	Yaw Value
Dz	Roll Value
Thrush	
Heading X	X-axis Airplane Heading
Heading Y	Y-axis Airplane Heading
Heading Z	Z-axis Airplane Heading
Position X	X-axis Airplane Position
Position Y	Y-axis Airplane Position
Position Z	Z-axis Airplane Position
Velocity	Airplane Velocity

Table-3. Format of output.csv for ATV Simulator.

Name	Definition
Speed	ATV Speed Value
Brake	ATV Brake Value
Steer	ATV Steer Value
Heading X	X-axis ATV Heading
Heading Y	Y-axis ATV Heading
Heading Z	Z-axis ATV Heading
Position X	X-axis ATV Position
Position Y	Y-axis ATV Position
Position Z	Z-axis ATV Position
Velocity	ATV Velocity

RESULTS

The data recording from output files (both from two simulators) show correct movements of all virtual position which are controlled by joystick or steer. Two output files, corresponding to the Airplane and ATV Simulators, are save in CSV. This formated file will be connected to the Massive Actuator Arrays platform.

Moreover, Figure-13 to the Figure-15 show the test results from Airplane Cockpit View Simulator, such as: from take-off position (pitch), Yaw movement and rotate to right or left (Roll).

Figure-13. Initial view and when the plane starts to move upwards (pitch).

Figure-14. Views when the aircraft straight and then turn right movement (Yaw).

Figure-15. Views when plane rotate to left or right (Roll).

In addition, the test results of the ATV cockpit view simulator can be seen also from Figure-16 through Figure-20. Those Figures show the ATV in several conditions, such as: going uphill, run sideways, braking mode, slip on ice, diving in the gravel-sandy road and muddy off-road.

Figure-16. Initial view and when the ATV drove on a flat road.

Figure-17. Views when the ATV going uphill and stopped because it forced to go uphill with 5th gear.

Figure-18. Views when ATV runs sideways and when braking.

Figure-19. Views when ATV running on ice and slipping when push the brake.

Figure-20. View when testing on gravel and sandy roads and on the muddy off-road area.

CONCLUSIONS

From the simulation results, it can be seen that the prototype cockpit view simulator for aircraft and ground vehicles ATV have been made correctly, according to the input controlled. Some environments, such as clouds and wind effects on aircraft as well as the effects of flat road, off road, ice, sand and mud on land vehicles, also have been added in the simulator. The two correspondence data in CSV format also have been created in order to connect both simulators to the Actuator Massive Array platform.

ACKNOWLEDGMENT

The authors would like to thank Directorate General of Higher Education for supporting him with the extended research fund under "Research Grant" No. 25/SP2H/PDSTRL_Pen/LPPM-UKP/IV/2014.

REFERENCES

[1] Pasila F. and Alimin R. 2013. Designing the 6-DOF Massive Parallel Arrays with Artificial Intelligence Control, Proceedings of the 2nd International Conference on International Conference on Robotic Automation System (ICORAS 2013).

[2] Adams E. 2010. Fundamentals of Game Design. Pearson Education, Inc.

[3] Ehrlich E. 1980. Oxford American Dictionary. Michigan: Oxford University Press.

[4] Pedersen R. E. 2003. Game Design Foundations. Wordware Publishing, Inc.

[5] Sauvé L., Renaud L., Kaufman D. and Marquis J. S. 2007. Distinguishing between games and simulations: A systematic review. Educational Technology and Society. 10(3): 247-256.

[6] Junker G. 2006. Pro OGRE 3D Programming. Appress. Springer-Verlag New York.

[7] OGRE3D Team. 2014. Object-Oriented Graphics Rendering Engine (OGRE 3D). Retrieved: 2014, march 17 http://www.ogre3d.org/.

[8] SDL Team. 2014. Simple Directmedia Layer (SDL). Retrieved: 2014 march 17. https://www.libsdl.org/.

[9] Bourg D. M. and Bywalec B. 2013. Physics for Game Developers 2nd. Ed. O'Reilly Media, Inc.

[10] Palmer G. 2005. Physics for Game Programmers. Apress. Springer-Verlag New York, Inc.

[11] NVidia PhysX Team. 2014. NVidia PhysX SDK. Retrieved: 2014 march 17. https://www.nividia.com/

[12] Palit A. K. and Popovic D. 2005. Computational Intelligence in Time Series Forecasting: Theory and Engineering Application. Springer-Verlag London.

[13] Palit A. K., Doeding G., Anheier W. and Popovic D. 2002. Backpropagation Based Training Algorithm for Takagi-Sugeno Type MIMO Neuro-Fuzzy Network to Forecast Electrical Load Time Series. FUZZ-IEEE'02: proceedings of the 2002 IEEE International Conference on Fuzzy Systems. Honolulu. Hawaii.