بسم الله الرحمن الرحيم

Sudan University of Science and Technology

College of Graduate Studies

Design of Virtual Reality Flight Simulator

تصميم محاكي طيران الواقع الإفتراضي

A dissertation submitted in partial fulfilment of the requirements for the award of Degree of Master of Science in Mechatronics Engineering

Prepared By:

Husham Hassan Badawi Zakaria

Supervised By:

Prof. Moutaman Mirghani Dafalla

(December, 2021)

DECLARATION OF THE STATUS OF THESIS

I hereby declare that this thesis is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at Sudan University of Science and Technology or other institutions.

Signature: _____

Name: Husham Hassan Badawi Zakaria

Date: December, 2021.

DECLARATION OF THE STATUS OF THESIS

I hereby declare that this thesis is based on the student original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at Sudan University of Science and Technology or other institutions.

Signature: _____

Name of Supervisor: Prof. Moutaman Mirghani Daffalla

Date: December, 2021.

ASSIGNING THE COPYRIGHT TO CGS, SUST

I, the signing here under, declare that I'm the sole author of the thesis for the Master of Science in Mechatronic Engineering entitled "DESIGN OF VIRTUAL REALITY FLIGHT SIMULATOR", which is an original intellectual work. Willingly, I assign the copy-right of this work to the College of Graduate Studies (CGS), Sudan University of Science and Technology (SUST). Accordingly, SUST has all the rights to publish this work for scientific purposes.

Candidate's Signature:

Candidate's Name: Husham Hassan Badawi Zakaria

Date: December, 2021.


قال تعالى
(اللَّهُ نُورُ السَّمَاوَاتِ وَالْأَرْضِ مَثَلُ نُورِهِ كَمِشْكَاةٍ فِيهَا
مِصْبَاحٌ الْمِصْبَاحُ فِي زُجَاجَةٍ الزُّجَاجَةُ كَأَنَّهَا كَوْكَبٌ دُرِّيٌّ
يُوقَدُ مِنْ شَجَرَةٍ مُبَارَكَةٍ زَيْتُونَةٍ لَا شَرْقِيَّةٍ وَلَا غَرْبِيَّةٍ يَكَادُ
زَيْتُهَا يُضِيءُ وَلَوْ لَمْ تَمْسَسْهُ نَارٌ نُورٌ عَلَى نُورٍ يَهْدِي اللَّهُ لِنُورِهِ مَنْ
يَشَاءُ وَيَضْرِبُ اللَّهُ الْأَمْثَالَ لِلنَّاسِ وَاللَّهُ بِكُلِّ شَيْءٍ عَلِيمٌ)
صدق الله العظيم
النور الآية (35)

DEDICATION

This thesis is dedicated to:

My homeland Sudan,

Sudan University of Science and Technology my second home,

My great parents, who never stop giving of themselves in countless ways,

My dearest wife, who leads me through the valley of darkness with light of hope and support,

My beloved kids: Mahmoud, Layal, and Mohammed whom I can't force myself to stop loving.

My colleagues who encouraged and supported me,

I dedicate this research.

ACKNOWLEDGEMENT

I would first like to thank my thesis advisor Prof. Moutaman Mirghani Dafallah from the National Centre for Research (NCR), director of Institute of Space Research and Aerospace (ISRA) who consistently allowed this thesis to be my own work, and steered me in the right direction whenever he thought I needed it.

Finally, I must express my very profound gratitude to my parents, wife and colleagues for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

ABSTRACT

Flight simulators are one of the most important factors influencing training because they play an effective role in reducing the cost and increasing the skills of the cockpit crew. There are several types of flight simulators depending on different criteria such as the visual arrangements, type of the platform weather fixed or has motion, and so many others. The full flight simulator (FFS) represents the highest level of simulation because of the meticulous details of all the aircraft parameters, yet it remains the highest in terms of cost. In this thesis something between the meticulous part and the cost side has been created by providing a virtual reality flight simulator. A 3D model of the aircraft cockpit ERJ-145 is constructed with all its internal instruments, display, indicators and switches in Blender application with the addition of other feature offered by Blender like skinning the cockpit and using animation as well. Then the cockpit is integrated in a visual engine namely Xplane v9 and tested for integrity. The results were quite acceptable since the plane worked and all instruments were alive during the flight, some criteria were referenced with the International Civil Aviation Organization (ICAO) for testing Flight Simulators. The simulator offers a new VR experience in flight simulation. The addition of other enhancement is recommended such as VR headset, haptic tracking, and other outer environment improvements. The simulator also could be linked to other simulator like an air traffic control (ATC) for more integration of training.

المستخـــلص

تمثل محاكيات الطائرات من أهم العوامل المؤثرة في التدريب وذلك لما تقوم به من دور فعال في تقليل التكلفة وزيادة مهارة الطاقم الجوي. هناك عدة أنواع من أجهزة محاكاة الطيران تصنف وفقاً لمعايير مختلفة مثل أنظمة الرؤية ونوع آلية الحركة سواء كانت ثابته أم متحركة، والكثير غيرها .يمثل جهاز محاكاة الطيران الكامل (Full Flight Simulator) أعلى مستوى من المحاكاة بسبب التفاصيل الدقيقة في صناعته، لكنه لا يزال أعلى من حيث التكلفة. في هذا البحث تم إنشاء شيء ما بين الدقة و التكلفة من خلال توفير جهاز محاكاة بإستخدام تقنية الواقع الافتراضي. لقد تم إنشاء نموذج ثلاثي الأبعاد لكابينة الطائرة ERJ-145 بكل العدادات وشاشات العرض المختلفة في برنامج التصميم Blender . كما تم إضافة مزايا آخري يوفرها البرنامج مثل تلوين الأجزاء وتحريكها. ثم تم دمج كابينة الطائرة في محرك بصري وهو Xplane v9 واختبارها . كانت النتائج مقبولة تماماً نظراً لأن الطائرة كانت تعمل وجميع العدادت كانت تعمل أثناء الرحلة كما تمت مقارنة الاداء مع بعض متطلبات المنظمة الدولية للطيران (الإيكاو) لتصميم المحاكيات. يوصى بإضافة تحسينات أخرى مثل نظارات الواقع الإفتراضي والتحسينات الأخرى للبيئة الخارجية. يمكن أيضًا ربط جهاز المحاكاة بأجهزة محاكاة أخرى مثل التحكم في الحركة لمزيد من تكامل في التدريب.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

No.	Symbol	Interpretation
1	a	Offset Coefficient
2	b	Scale Coefficient
3	α	Yaw Counterclockwise Rotation
4	β	Pitch Counterclockwise Rotation
5	γ	Roll Counterclockwise Rotation
6	R_{eye}	The View Point Of Orientation
7	$\hat{\omega}$	Estimated Angular Velocity
8	ω	True Angular Velocity
9	$\hat{\theta}$	Estimated Orientation
10	θ	True Orientation
11	k	Sampling Rate
12	M	Model's Orientation Matrix
13	m	Any Real Number
14	U	Horizontal Coordinate of a Mesh
15	V	Vertical Coordinate of a Mesh
16	V2	Take-off Safety Speed

LIST OF ABBREVIATIONS

Abbreviation	Interpretation
2D	Two-Dimensions
3D	Three-Dimensions
AC	Alternating Current / Aircraft
AR	Augmented Reality
ATC	Air Traffic Control
CRT	Cathode Ray Tube
DC	Direct Current
DHM	Dexterous Hand Master
DIY	Do It Yourself
DLP	Digital Light Processing
DOF	Degrees of Freedom
EASA	European Aviation Safety Agency
EICAS	Engine Indication and Crew Altering System
EOG	Electrooculography
ERJ	Embraer Regional Jet
FS	Flight Simulator
FSTD	Flight Simulation Training Device
FFS	Full Flight Simulator
FOV	Field of View
FPS	First-Person Shooter / Frame Per Second
FFS	Full Flight Simulator
GI	Global Illumination
GPU	Graphical Processing Units
G/S	Glide Slope
HMD	Head Mounted Display
Hz	Hertz
ICAO	International Civil Aviation Organization
IMU	Inertial Measurement Unit
iOS	iPhone Operating System
KIAS	Konts Indicated Airspeed
LAN	Local Area Network
LCD	Liquid Crystal Display
LCoS	Liquid Crystal on Silicon
LED	Light-Emitting Diode

MFD	Multi-Function Display
MIT	Massachusetts Institute of Technology
MR	Mixed Reality
OpenGL	Open Graphics Library
P3D	Prepar3D
PC	Phase-Coherent / Personal Computer
PFD	Primary Flight Display
RVR	Runway Visual Range
SGI	Silicon Graphics International
TOF	Time-of-Flight
TUM	Technische Universität München
UE	Unreal Engine
USAF	United States Air Force
USB	Universal Serial Bus
VATSIM	Virtual Air Traffic Simulation
VE	Virtual Environment
VR	Virtual Reality
VRFS	Virtual Reality Flight Simulator
VVI	Vertical Velocity Indicator
VWG	Virtual World Generators
WoW	Window on World

CHAPTER ONE

INTRODUCTION

CHAPTER ONE

INTRODUCTION

1.1 Preface

If aerospace is one of the finest modern sciences, then flight simulation is one of the most complexed. The invention of the airplane by the Wright brothers was a major milestone in the scientific and technological applications in our modern history so as flight simulators.

The first issue that emerged in the beginnings of aviation is how to build an airplane in a way that makes it fly, and the second is to find ways to train flying those airplanes properly. Over time, performance manuals and technical data have become more complexed, thus increasing the burdens on pilots, not to mention the increased complexity of aircraft systems and the need for pilots to keep updated on a regular basis which has exacerbated the issue. To simplify this, flight simulators emerged as a response to simplifying and solving those problems.

Flight Simulators are designed to be used for different purposes, for example pilots and engineers training, which contributes to some extent to efficient and qualified personnel. Full flight simulators (FFS) are well known for the high ability to mimic real environment and interaction, but on the other hand it has a very high amount of cost due to that fact that it is designed only for specific type of aircraft or helicopter or sometimes an entire aircraft family.

1.2 Problem Statement

The main problem of flight simulators is the high cost of establishing such devices, nevertheless low-cost flight simulators were also introduced but it affects the quality of training. Also, FFS are cockpit dependent upon. Generic FFS were also developed for lowering the cost of several simulators but again this has major implications on both cost and quality.

1.3 Proposed Solution

Designing a generic virtual reality flight simulator capable of simulating an airplane with complete flight physics, dynamics and surrounding environment which can be used in airmen and engineers training. Thus, in return lowering the cost of building a flight simulator as operation costs with regards to quality.

1.4 Objectives

- To design a flight simulator that contributes to lower the cost of building such establishments.
- To not affect the quality of training by reducing the cost of building.
- To integrate the new technology emerging i.e. virtual reality in the design of flight simulators.

1.5 Methodology

A comprehensive study is conducted to conceptualize virtual reality with flight simulators. Then the type of the aircraft is selected, designed in a 3D model, Afterwards the 3D model is placed inside a visual engine to run the model.

1.6 Research Outlines

Chapter one provides an introduction to the research, the problems statement, proposed solution, objectives and the research methodology.

Chapter two is a review of literature of the virtual reality (VR), including scientific papers of different authors in the field of VR Technology. As well as the history of flight simulator and the stages of its development.

Chapter three provides an extensive insight of view VR technology, how does it work, related terms and definitions, and the interaction of the VR with human physiology.

Chapter four is the implementation of VR into flight simulator, designing different parts of the simulator, and the mechanism of work.

Chapter Five discusses the findings of this research and provides relevant recommendations for future work.

CHAPTER TWO

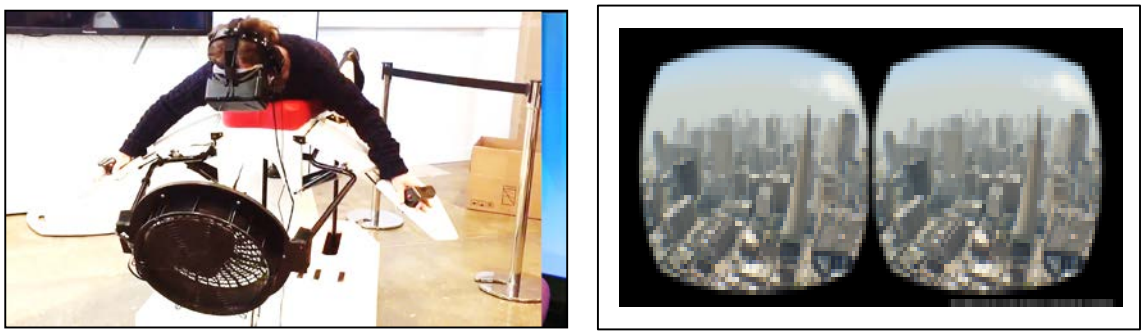
BACKGROUND AND LITERATURE REVIEW

CHAPTER TWO

BACKGROUND AND LITERATURE REVIEW

2.1 Background

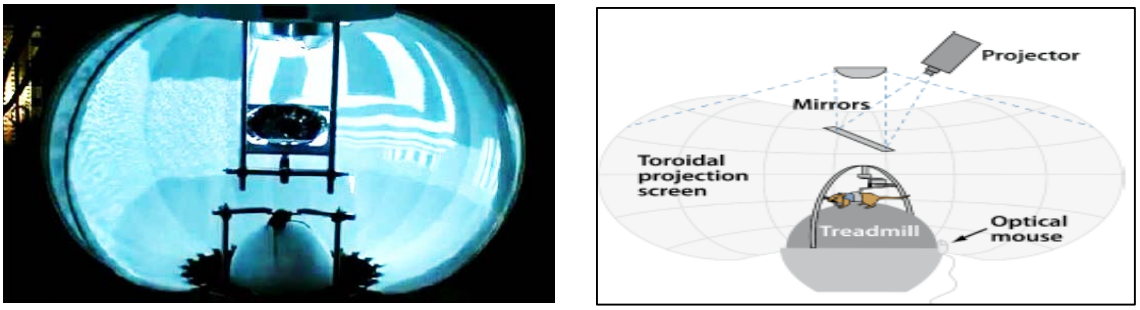
Normally in the beginning Virtual Reality (VR) is defined, but before that, these two examples of VR experiences are explored, the first example as shown in figure 2-1: (a) illustrates a man flapping his arms facing an air blower, this resembles a bird flapping its wings and as the person moves his arms the air hits the person and views a virtual city while flying and flapping, the scene is clearly visible in figure 2-1 (b) where the user sees the city of Zurich as a bird eye view.



(a) (b)

Figure 2-1: Bird Experiment from Zurich University of Arts [1]

The other example in figure 2-2: (a) shows a rodent running on a spherical ball that acts as a treadmill while viewing virtual maze, as seen in the configuration in figure 2-2 (b) the use of projectors and collimated establishments, such establishments are used to investigate the neural foundations of behavior, the use of VR technology allows these types of investigations which cannot be easily used with classical behavior setups.



(a) (b)

Figure 2-2: A Rodent Experiencing Virtual Maze [1].

Thus, there are many definitions for VR, but we wanted our definition to be broader enough to cover commonalities and differences as well as merits of the mentioned examples, so VR might be defined as: The illusion of participation in an artificial sensory simulation rather than external observation.

2.2 VR, AR, MR, and VE

Computer Graphics basically means creating and manipulating images to produce interactive images, animations, etc. Use of computer graphics led to development of Augmented Reality, Virtual Reality and Mixed Reality [2].

There is some difference between Virtual reality (VR), Augmented Reality (AR), and Mixed Reality (MR), whereas VR is the use of computer graphics systems in combination with various display and interface devices to provide the effect of immersion in the interactive 3D computer-generated environment. We call such an environment a virtual environment (VE) [3].

Mixed reality (MR) refers to the incorporation of virtual computer graphics objects into a real three-dimensional scene, or alternatively the inclusion of real-world elements into a virtual environment. The former case is generally referred to as Augmented Reality (AR) [3].

The objective of an AR framework is to improve the client's impression of and cooperation with this present reality by supplementing this present reality with 3D virtual articles that seem to exist together in an indistinguishable space from this present reality [2].

2.3 Literature Review

2.3.1 History of VR

The very first idea of VR was presented by Sutherland in 1965; where he stated: make that (virtual) world in the window look real, sound real, feel real, and respond realistically to the viewer's actions, but however, history does not really begin there.

It was early science fiction literature that sparked the imaginations of inventors to try to recreate these artificial or illusory environments with technology. Science fiction literature has even coined and/or popularized some of VRs most used terms, such as cyberspace or avatar [4]. Burdea and Coiffet in 2003 described VR as an integrated trio of immersion, interaction, and imagination i.e. the three I's of VR [5].

Probably the first VR device that encompassed all three I's was Morton Heilig's Sensorama seen in figure (2-3), he wanted to create a technology that he called cinema of the future the basic concept of which was a technology of total immersion into a film in which viewer will not only see the image and hear the sound, but also would experience other physical sensations – the smells, the shaking, the wind, etc.

Initially, Heilig called his creation the theater immersion, but later, when the time came to patent the development, the project was given the name Sensorama. Unfortunately, work on the project was frozen soon as

Heilig hasn't received any financing, so crucial for the further development of the project. The investors just could not understand that such a technology could have been successfully sold.

However, Heilig was tagged as the father of VR, and his Sensorama became, in a sense, the prototype of the future 3D-cinemas and attractions. Heilig also received a US patent for his Head Mounted Display (HMD) that supported stereo sound and an odor generator as shown in figure (2-4) [4].

The 1960s and 1970s continued to see significant advancements in VR and computer graphics. One of the most notable would be in 1963 when Ivan Sutherland's published his dissertation at Massachusetts Institute of Technology (MIT) on the first interactive computer graphics system named Sketchpad.

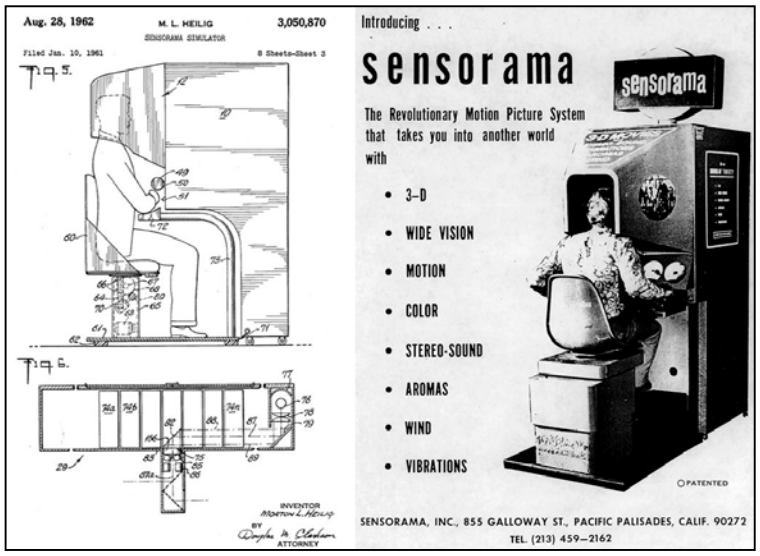


Figure 2-3: Morton Heilig's Sensorama [1]

The Sketchpad system makes it possible for a man and a computer to converse rapidly through the medium of line drawings. Heretofore, most interaction between man and computers has been slowed down by the need to reduce all communication to written statements that can be typed [6].

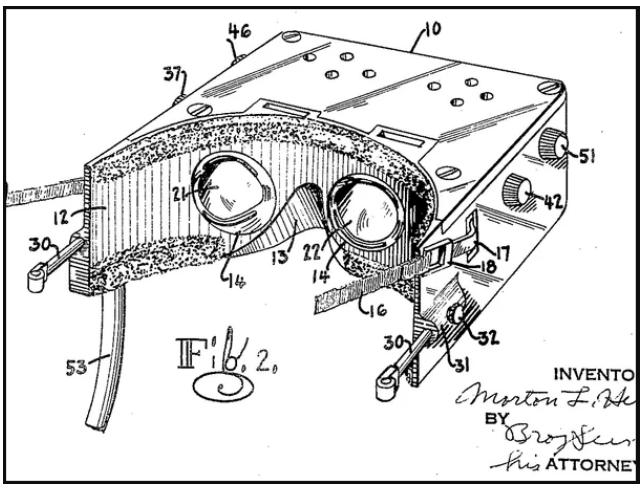


Figure 2-4: Morton Heilig's HMD [7]

If we further investigate the history of VR, we can find aspects in earlier history that contributed to some extend to the current VR; the paintings in the walls more than 30,000 years ago leaving so much to a human to imagine, putting pictures in motion also considered part of this history as well.

If we recall the first experience in the cinema where a train was moving towards the spectators fooling them, it was going to hit them although it hadn't any audible sound for more realism but actually it worked. The next era was the animated pictures or some might refer to it as anime or cartoons.

Unlike motion pictures or cartoons video games as a great interaction capability in terms of closed loop interaction; where the difference here between closed loop and open loop is that if a person has partial control over the sensory simulation then this is considered as closed loop and vice-versa, the closed loop control varies in regards to person motion like movement of eyes, hands, heads and others.

Different video games we can browse here as part of this heritage such as the famous Super Mario Bros which represented a third person perspective. Where a first-person shooter games (FPS) such as Call of Duty as seen in figure (2-5).

Figure 2-5: Call of Duty Video Game [7]

Additionally, a device was introduced by Sir Charles Wheatstone in 1838 called the stereoscope that used mirrors to depict different image to left and right eyes to induce a 3D effect, this system employs two cameras and two projectors, and, furthermore, requires a spectator to use polaroid glasses in order to have the left eye image reaches only the left eye and the right eye image reaches only the right eye.

Although this system does provide true 3D, it is hampered seriously by the fact that only one-twelfth of a viewer's field of vision is used. Therefore, objects floating in space are disagreeably truncated by the pictures frame [1], thus another way to increase the sense of immersion and depth was to increase the field of view. The Cinerama system from the 1950s offered a curved, wide field of view that is similar to the curved, large LED (Light-Emitting Diode) displays offered today, along these lines, we could place screens all around us.

This idea led to one important family of VR systems called the CAVE, which was introduced in 1992 at the University of Illinois, the user enters a room in which video is projected onto several walls. The CAVE system also offers stereoscopic viewing by presenting different images to each eye using polarized light and special glasses. Often, head tracking is additionally performed to allow viewpoint-dependent video to appear on the walls [1].

2.3.2 Flight Simulators

Before the introduction of flight simulators, a pilot learnt how to fly by instruction from another pilot, flight simulators were designed to be used for pilots training and engineers training, which contribute to some extent to efficient and qualified personnel. Full flight simulators are well known for the high ability to mimic real environment and interaction, but on the other hand it has a very high amount of cost due to that fact that it is designed only for specific type of aircraft or helicopter or sometimes aircraft family.

Flight Simulators or full-scale flight simulators are being used since the past century and they played great role in rising up the efficiency of ground and air crew, the evolution of flight simulator has taken wide strides towards more realistic simulations in regards to visual system, reactions of the aircraft or the engines, or emulating the motion of the aircraft until reaching 6 degrees of freedom (6DOF). Some flight simulators serve more than one type or similar types as this one is called generic. Full Flight Simulators (FFS) has an exact replica of the cockpit.

2.3.2.1 History of Flight Simulators

By 1910, the primitive means of flight training took the initial form, a mockup of an aircraft consists of two halves of a barrel, where the bottom half consists of the base and the upper section represents the cockpit of the

aircraft, and connected with pulleys and wires with controls to resemble the process of keeping the balance of the real plane.

In 1927, Ed Link invented his own flight training device when he was unsatisfied by the way the training was held. The trainer was based on the vacuum technology used in automatic musical instruments of the 1920s. In fact, the earliest trainer sat on a series of organ bellows, which would inflate or deflate to various heights to cause the trainer to bank, climb, and dive. In 1930, Ed Link organized the Link Flying School in Binghamton, New York. The trainer allowed him to reduce the cost of flying lessons by providing a way for the student pilots to learn some flying skills on the ground [7].

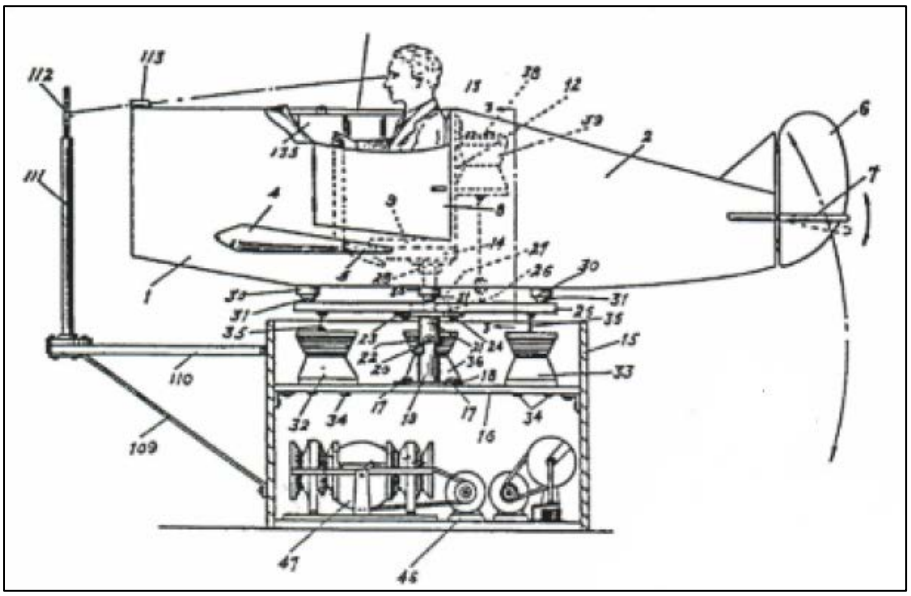


Figure 2-6: Edwin Link Drawing of the Simulator for Patent Application

After the Link's electromechanical training device, immediately after World War II, simulators based on standard primitive computers emerged and the computer began to take its share in facilitating and improving data for flight simulators. In 1948, Curtis Wright delivered the first flight simulator to a civilian airliner, Pan American, which was not mobile but static, not even with visual systems for displaying the aircraft's external

environments, but it provided an excellent opportunity to understand the aircraft's systems.

Movable platform flight simulation systems emerged in the late 1950s, and were equipped with a terrain display system that was limited in width and limited to the area adjacent to the airport to be operated. As if they were magnified, so if a very short flight is to be done, it would require a model the size of a giant sports stadium.

In the 1960s, digital computers were first used in flight simulators, and by the seventies, motion platform with six hydraulic rods were integrated in the design of flight simulators. It should be noted that the flight simulators are also subject to licensing to be certified as an aviation training device, just like aircraft.

There is no doubt that flight simulators are indispensable. As the computer and optical systems evolve, pilots' sensation will be improved through the virtual reality generated by these devices. Flight in the simulator is usually harder than actually flying the plane itself, as we have pointed out because it was designed to do so, in order to simulate the hardest conditions but without loss of life.

2.3.2.2 Virtual Reality Flight Simulators

Virtual reality flight simulators (VRFS) was an attempt to reduce the cost implemented by using high-end technology and lots of hardware in order to make the experience of the flight real. Aslandere from Technische Universität München (TUM) in 2018, they presented an outline of a generic distributed virtual reality application which is aimed to meet the needs of the aerospace industry [7]. The preliminary results show that the VRFS is a

promising flight simulator concept in spite of the real time constraint. The VRFS is used as an engineering flight simulator for testing new aircraft concepts at the moment. The virtual hand-button interaction might be sufficient for virtual prototyping but it is not ready for pilot training yet.

Valentino in 2017 developed a virtual reality flight simulator successfully to simulate the flying of the airplane namely Cessna 182 with simple flight dynamics, limited terrain and objects [8]. It gives great perspective of flying in mid-air, they used Samsung Galaxy S7 with Android Marshmallow v6.0.1 and virtual reality supported Samsung Galaxy Gear VR, Gamepad, laptop, and a unity software.

CHAPTER THREE

VIRTUAL REALITY TECHNOLOGY

CHAPTER THREE

VIRTUAL REALITY TECHNOLOGY

3.1 Components of VR

As any computerized system VR consists of two main components; hardware and software components.

3.1.1 Hardware

The hardware produces stimuli that override human senses; in return hardware is subdivided into: Input devices and sense organs, Computer workstation, and output devices or displays. Figure (3-1) shows the arrangement of hardware among other components in a generalized VR system.

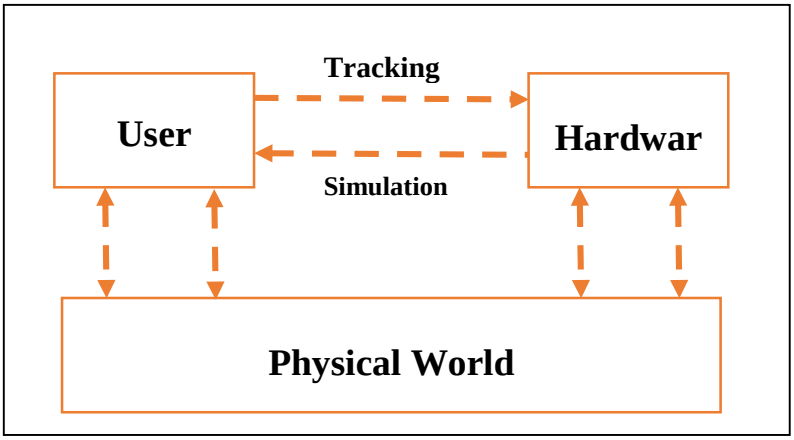


Figure 3-1: Placement of Hardware in a VR System

3.1.1.1 Input Devices, Sensors, Sense Organs and Tracking Devices

Three categories of tracking may appear in VR systems, based on what is being tracked; the first category is the user's sense organs: most of the focus is on head tracking, which is sufficient for visual and aural components of VR; however, the visual system may further require eye tracking if the

rendering and display technology requires compensating for the eye movements [1].

The second category is the user's other body parts: if the user would like to see his body parts in VE then tracking of the user body parts is crucial. Perhaps facial expressions or hand gestures are needed for interaction. And the third category is the rest of the environment which is the real world that surrounds the user.

Position and orientation tracking devices provide information to immersive VR system, for instances the position and orientation of the user's head required for rendering of the images, also other body parts can be tracked to render images according to their relative movement.

Three-dimensional objects have six degrees of freedom: position coordinates (x, y, and z offsets) and orientation (yaw, pitch, and roll angles for example). Each tracker must support this data or subset of it. In general, there are two kinds of trackers: those that deliver absolute data (total position / orientation values) and those that deliver relative data (i.e. a change of data from the last state) [9].

3.1.1.1.1 Tracking 2D Orientation

The 2D orientation of a rigid body is estimated using Inertial Measurement Unit (IMU); so, the application is determining the view point of orientation (R_{eye}), and determining the orientation of a hand-held controller. In fact, ever body part or moving object in the physical world can be determined if has an IMU attached to.

The orientation of 2D is the basis and the concept will further be applied to 3D one. First, let's imagine mounting a gyroscope on spinning

merry-go-around or carousel to measure the angular velocity. In this case the gyroscope will be producing an estimated measurement of angular velocity denoted by $\widehat{\omega}$, while the true value of the angular velocity denoted by ω, because of the calibration error where $\omega \neq \widehat{\omega}$ unless in ideal conditions; different IMUs correspond to different accurate measurements, thus:

$$\widehat{\omega} = a + b\omega \quad (3\text{-}1)$$

Where: $a\ and\ b$ are offset and scale coefficients respectively, if the measurement is 100% accurate then $\omega = \widehat{\omega}$ and $a = 0, b = 1$. Now combining the measured and the true angular velocities we have:

$$\widehat{\omega} - \omega = a + b\omega - \omega = a + \omega\,(b - 1) \quad (3\text{-}2)$$

Let's assume we've used a sensor to estimate the orientation of the merry-go-around, so we'll compute the estimated orientation $\hat{\theta}$ and compare it to the true value of θ. By using this imperceptibly over time will produce the drift error denoted by $d(t)$ which is:

$$d(t) = \theta(t) - \hat{\theta}(t) \quad (3\text{-}3)$$

For simplicity suppose $\theta(0) = 0$ and ω is constant, by integrating (3-2) the drift error is:

$$d(t) = (\widehat{\omega} - \omega)t = (a + b\omega - \omega)t = (a + \omega(b - 1)t \quad (3\text{-}4)$$

The drift error is proportional to $a\ and\ b$ as a deviates from 0, and b from1.

Considering the second component and ignoring a will result in the drift error proportional to the speed of the carousel. In VR headset with a gyroscope; means the rate of the tracking error increases as the user's head rotates quickly.

We've four major problems if dealt with correctly will have an effective tracking system, they are:

1- Calibration: if we've two sensors and one of them is accurate, then the other can be calibrated to work closely to the other.

2- Integration: The orientation is determined by aggregating the measurement of discrete points in time provided by the sensor, and the output of the sensor arrives at a regular sampling rate, for instance suppose we've a sensor characteristic; provided measurement time 1 ms = 1000 Hz sampling rate.

Let $\hat{\omega}|k|$ denotes to the K_{th} sample which arrives at $k\boldsymbol{\Delta t}$, we can estimate $\theta(t)$ at $t = k\boldsymbol{\Delta t}$, by integration;

$$\hat{\theta}|t| = \theta(0) + \sum_{i=1}^{k} \hat{\omega}\,|i|\,\boldsymbol{\Delta t} \qquad (3\text{-}5)$$

Each $\hat{\omega}\,|i|$ rotates $\Delta\theta(t) = \hat{\omega}\,|i|\,\boldsymbol{\Delta t}$, we can rearrange the equation as:

$$\hat{\theta}|k| = \hat{\omega}|k|\,\boldsymbol{\Delta t} + \hat{\theta}[k-1] \qquad (3\text{-}6)$$

3- Registration: The initial orientation must be determined which is the initial alignment between the real and virtual world.

4- Drift error: which grows overtime and cannot be allowed to accumulate.

3.1.1.1.2 Tracking 3D Orientation

The gyroscope measures angular velocities along three orthogonal axis which results in $\hat{\omega}_x.\,\hat{\omega}_y.\,and\ \hat{\omega}_z$, the sensing elements in the gyroscope are micromachined mechanical that vibrate and if the sensor rotates in its direction of sensitivity then the elements output is converted to electrical signals, in turn the electrical signals are calibrated to produce an output in degrees or radians per seconds.

IMU's are discussed before have commonly acetometers that measure linear acceleration along the three axes to obtain $\hat{a}_x.\,\hat{a}_y.\,and\ \hat{a}_z$. Likewise, in the 2D tracking, the elimination of calibration, integration, registration, and drift error is essential.

3.1.1.1.3 Magnetic Trackers

There are two varieties of magnetic tracker in today VR application, one uses alternating current (AC), and the other one uses direct current (DC).

The alternating current magnetic tracker is composed of three units: a magnetic emitter assembly, a magnetic receiver assembly, and a control unit. The emitter assembly is constructed of three mutually perpendicular coils that emit a magnetic field when fed a current. The sensor assembly is, likewise, constructed of three mutually perpendicular coils that produce currents when moved through the magnetic field, a current is sent to the emitter coils in a sequence that radiates three mutually perpendicular nutating (rotating) magnetic fields.

The field induces currents in the sensor coils; the current induced in each coil varies with the distance from the emitter. Sensor position and orientation are calculated from these nine induced currents (three sensor currents for three emitter coils in each nutation). Position and orientation are determined by calculating the small changes in the sensed coordinates and then updating the previous measurements [9].

Direct current emitters represent the recent development in the field; these systems allow the sensor to work closer to metal objects better off the alternating current trackers.

The emitter radiates a sequence of dc pulses; in effect switching the emitted field off and on. This design is intended to reduce the effect of distorting eddy currents induced by metallic objects because eddy currents are created only when the magnetic field is changing [9].

Figure 3-2: Emitter and Receiver Units of a Magnetic Tracker [9]

3.1.1.1.4 Acoustic Trackers

Also known as ultrasonic trackers since they use waves above 20 KHz for determining the position and orientation of objects. Using sound allows only determination of distance between two points; hence sets of emitters and receivers are used to determine accurate position and orientation.

The two types of acoustic trackers either they use Time-of-Flight (TOF) or Phase-Coherent (PC). TOF trackers measure the time of pulses from emitter to receiver, whereas PC trackers compare the phase of a reference signal with the phase of the received signal in calculating the position and orientation.

3.1.1.1.5 Optical Trackers

Optical Trackers are classified into three categories:

The first category is Beacon Trackers: they use a bevy of beacons and cameras to capture the beacons patterns and since the geometries of the beacons and cameras are known it is relatively easy to calculate the position and orientation.

The second category is Pattern Recognition: these systems determine position and orientation by comparing known patterns to sensed ones.

The third category is Laser Ranging: This approach uses laser transmitted onto the tracked objects through diffraction grating. A sensor is implemented to analyze the diffraction pattern to calculate the position and orientation.

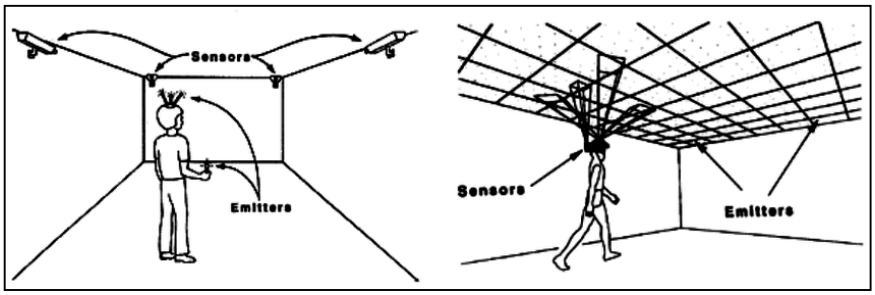


Figure 3-3: Beacon Trackers [9]

3.1.1.1.6 Mechanical Trackers

A group of mechanical arms with joints are used to measure position and orientation of a free joint in relation to the base. The angles of the tracker are measured with the help of potentiometers. This allows deriving the position and orientation.

3.1.1.1.7 Eye Tracking

From the user's point of view eye tracking is the most appropriate way of image rendering; the most important eye tracking technology is discussed as follows:

1- Limbus tracking: the sharp boundary between the iris and the sclera (limbus) can be easily identified. The infrared LEDs and photo-transistors are mounted on the user's glasses to monitor infrared spots reflections from the iris and sclera in order to determine the gaze direction. This technique offers good accuracy (1° to 3°), but limits vertical eye movements (by extreme vertical eye movements limbus is partially obscured by eye-lids what hinders exact measurement) [9].

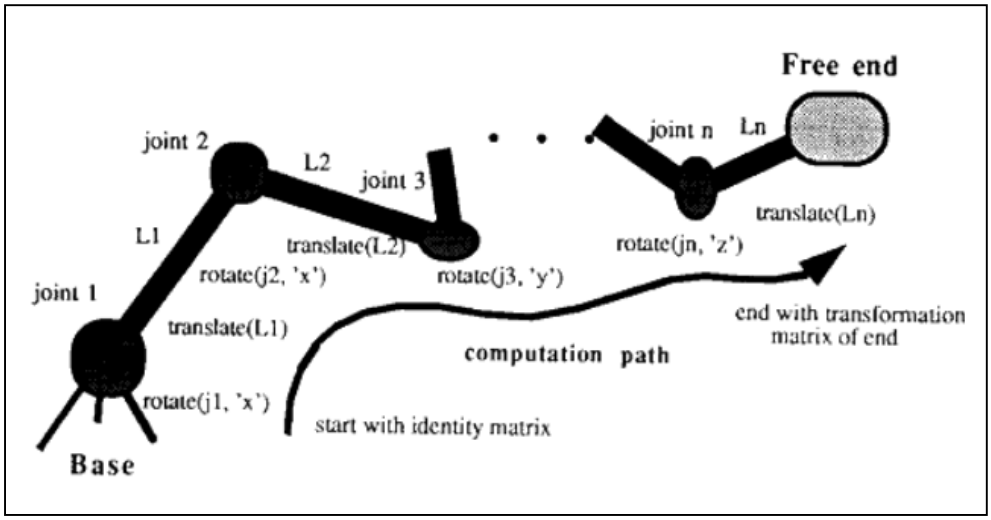


Figure 3-4: The Principle of Mechanical Trackers [9]

2- Image tracking: this technique uses video camera and image processing in order to determine the gaze direction.

3- Electrooculography (EOG): this one used to measure the potential difference between the front and the back of the human eye caused by electrodes placed besides the eyes to measure the corneo-retinal standing potential that exists.

4- Corneal reflection: a beam is transmitted to the surface of the cornea then the reflected one is analyzed in terms of photo-transistors functionality.

3.1.1.1.8 3D Input Devices

These devices were developed to facilitate the human-computer interaction; they may be either attached to our bodies or hand-held. The functions of these devices are to select, move, modify, reposition, etc of virtual objects.

3D Mice and Bats: They somewhat represent the basic and simple devices or interaction tools that one can use. Also they are equipped with buttons for other purpose.

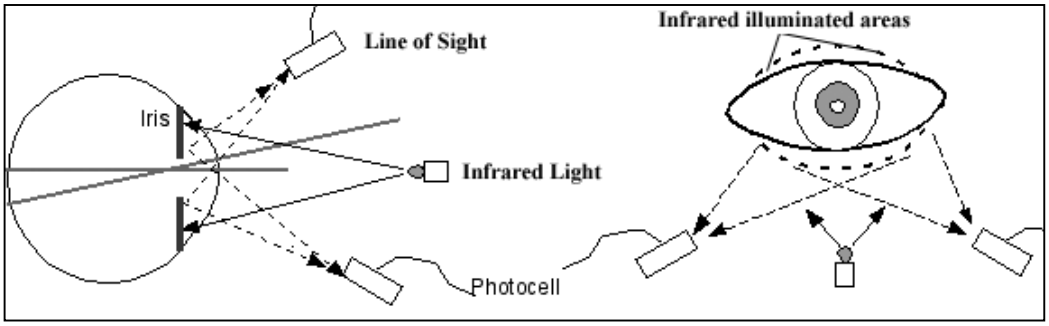


Figure 3-5: Limbal Tracking System [1]

Gloves: Mainly they detect the joint angles of finger by using fiber optic sensors, foil-strain technology or resistive sensors; additionally, they are enhanced with a tracker for better calculations of position and orientation.

Dexterous Manipulators: They are developed for applications of VR that require precise control; for instances, surgical operations. Further development of these devices is the Dexterous Hand Master (DHM) they can trace three joint angles for every finger i.e. 4DOF for every finger, and 20DOF for one hand.

3.1.1.1.9 Desktop Input Devices

What makes these devices popular is the low-price and functionality rather than the 3D input devices, although they decrease the level of immersion they still handy.

Space Ball: It provides 6DOF, that the user holds the ball and does the manipulation. It has buttons that do special functions for more interaction.

Cyber Man: Same as the space ball it provides 6DOF, it is the most used control in computer games, and it also sometimes comes with vibration motor to indicate an unusual event or attention grabbing property.

2D Input Devices: These 2D devices are popular, widely separated, and relatively cheap, that does not mean less efficiency nevertheless they are used

to control 3D objects. The philosophy behind this is the design of a virtual controller to be controlled by these 2D devices; in return the virtual controller can command the 3D object.

Figure 3-6: Dexterous Hand Master [1]

3.1.1.2 Computer Workstation

The main function of computer workstations is to execute the virtual world generators (VWG) and they are normally higher than regular personal computers (PCs). They offer great capabilities for visualization and manipulation of graphics, example of these workstation sold by Dell, HP, and Silicon Graphics International (SGI).

In addition to the main computing systems, specialized computing hardware may be utilized; graphical processing units (GPUs) have been optimized for quickly rendering graphics to a screen and they are currently being adapted to handle the specific performance demands of VR. Also, a display interface chip converts an input video into display commands. Finally, microcontrollers are frequently used to gather information from sensing devices and send them to the main computer using standard protocols, such as USB [9].

Virtual world generators run on the computer workstation and as the name implies are generating the virtual world which might be synthetic world, previous record of the real world, or a live connection to another part of the real world. The user discerns the generated virtual world through the targeted sense organ using a display designed to mimic the type of stimulus that appears without VR.

In the case of human eyes, the display might be a smart phone screen or the screen of a video projector. In the case of ears, the display is referred to as a speaker. A display need not be visual, even though this is the common usage in everyday life [1]. The process of extracting the visuals from the VWG to the display system is called rendering, it will be discussed later in this chapter.

3.1.1.3 Displays

They are used to generate a stimulus for a targeted sense organ, and since vision contributes up to 70% of the human sense stimulation in VR, the main concentration will be for the eye. There are many implementations of displays in VR applications; for cave systems projectors and mirrors are used, different types of projectors technology are into usage, including DLP (Digital Light Processing), LCD (Liquid Crystal Display), and LCoS (Liquid Crystal on Silicon). For headsets, smart phones' screens are used as displays and are put close to eyes with the help of lens for each eye.

Currently screen manufacturers are focusing on screens for headsets by leveraging the latest LED display technology from the smart phone industry, some are targeting one display per eye with frame rates above 90Hz and over two megapixels per eye [1].

Displays are also for other sense organs of the human body i.e. for the hearing sense organ; speakers address the ear, also bone conduction is used which a vibration is generated into the skull making the bones conduct this vibration to the inner ear. For touch, there are haptic displays. Haptic display is given in some sort of vibration, pressure, or temperature. An example of this haptic feedback is the deployment of vibration in the controller whenever unusual event occurs.

3.1.2 Software

There are two ways in order to program a VR system, either by providing high-level description which is a pre-programmed VR engine automatically that specifies all the low-level details, but this method is far to reach having a fully general functional engine.

The good news is that VR engines are likely to emerge towards specialized engines, for instance an engine is targeting cinema immersion, while another engine is targeting engineering designs. The other way around is to develop the software from scratch; in returns this requires a deeper understanding of VR systems and deeper knowledge with the low-level systems. The advantage of developing the software from scratch is the ability of the programmer to execute ideas not possible in dedicated engines.

As mentioned in section 3.1.1.2 that computer stations execute VWG which simply represent the software discussed above. The key role of the VWG is to maintain enough of an internal “reality” so that renderers can extract the information they need to calculate outputs for their displays [6].

We will discuss in greater details the engines or maybe known as game engines; which is a software framework designed for the creation and

development of video games, and virtual environments. Developers use them to create games for consoles, mobile devices and personal computers.

The core functionality typically provided by a game engine includes a rendering engine (renderer) for 2D or 3D graphics, a physics engine, a collision detection (and collision response) system, sound, scripting, animation, artificial intelligence, networking, streaming, memory management, threading, localization support, scene graph, and may include video support for cinematics. The process of game development is often economized, in large part, by reusing/adapting the same game engine to create different games, or to make it easier to "port" games to multiple platforms [10].

3.1.2.1 Computer Graphics

Any graphical image that being viewed or represented on a computer monitor is called computer graphics, also there is another related term which is image-processing; the difference between the two is that computer graphics generates its own image, and image-processing are captured by a camera or any other capturing device.

Computer generation of photorealistic graphics can be separated into two distinct parts; modelling and rendering. Modelling involves creating objects, moving them around to arrange a scene, defining how each object will look in the scene, and defining how the lighting and camera will look in the scene. Rendering involves making a realistic image out of the modelled scene by applying surface characteristics to the surfaces of the objects in the scene. [10].

3.1.2.1.1 3D Modeling

Every computer-rendered image requires three essential components: a 3D scene description, one or more sources of light, and a description of the camera or eye viewing the scene. The scene description is typically composed of one or more models, or 3D structures. Typically, we think of a model as a stand-alone part, e.g. a pencil or a tree, and the scene as the assembly of these parts into a complete 3D environment. This attitude reflects the most common procedure for building up a 3D scene: one builds many models, and then assembles them [11].

If we work with high-level engines to build a VR experience, then most of the following concepts might not seem necessary. You can just select options from the engine with simple codes to pull everything together. However, an understanding of the basic transformations is essential to making the software do what you want. Furthermore, if you want to build virtual worlds from scratch.

Firstly, we need a virtual world to contain our model, let's denote this virtual world with $\Re^3$ in which every point is represented as a triple of real-valued coordinates: (x, y, z). Models in the virtual world are composed of infinite number of points and are defined in terms of primitives in which each represents an infinite set of points. The simplest and most useful primitive is a 3D triangle, as shown in figure 3-7, all of the triangle is being represented by the coordinates of the triangle vertices:

$$((x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3)) \quad (3\text{-}7)$$

To model a complicated object or body in the virtual world, numerous triangles can be arranged into a mesh [6], but let consider we listed all triangles that forms mesh into an array or memory and most of the triangles share the vertices, so clearly there will be redundancy in the memory. To overcome this problem, we can use the doubly connected edge list, also known as half-edge data structure.

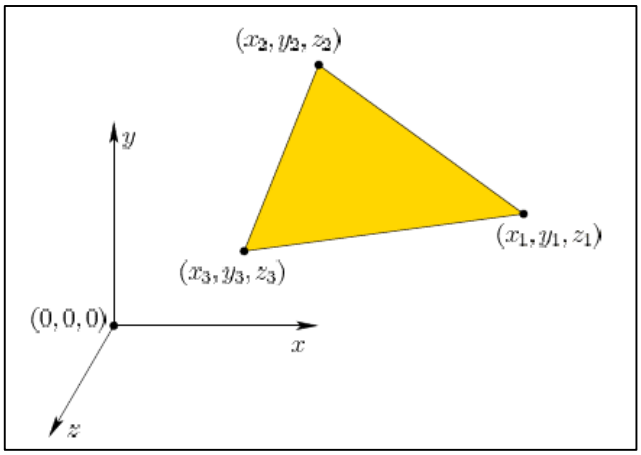


Figure 3-7: Points in the Virtual World [1]

In this method there are three data elements namely, faces, edges, and vertices. These represent two, one, and zero-dimensional parts, respectively, of the model. In our case, every face element represents a triangle. Each edge represents the border of one or two triangles, without duplication. Each vertex is shared between one or more triangles, again without duplication [6].

We've chosen triangles to represent the model in the virtual world because they are simple to handle in terms of algorithms, especially if implemented in hardware. We could've chosen other primitives such as such as quadrilaterals, splines, and semi-algebraic surfaces but this could lead to smaller model sizes, but often comes at the expense of greater computational cost for handling each primitive.

Indeed, there are two types of models; stationary models which tends to be fixed in the same coordinates such as: streets, buildings, and floors. The other type is the moveable models which transform in terms of position and orientation such as vehicles and furniture.

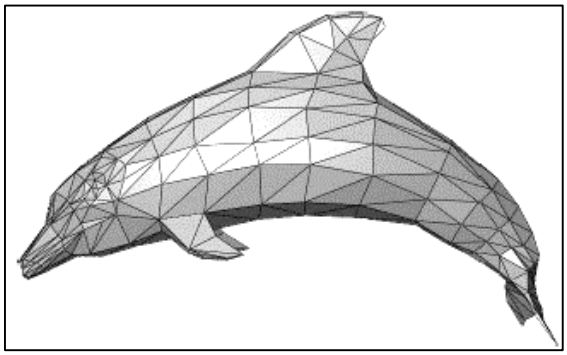


Figure 3-8: A Dolphin Created Using Triangular Mesh [9]

Motion can be caused in a number of ways. Using a tracking system, the model might move to match the user's motions. Alternatively, the user might operate a controller to move objects in the virtual world, including a representation of himself. Finally, objects might move on their own according to the laws of physics in the virtual world [6].

The operation of changing position is called translation, t is the amount we want to change and it is given by:

$$(x_1, y_1, z_1) \longrightarrow (x_1 + x_t, y_1 + y_t, z_1 + z_t) \quad (3\text{-}8)$$

$$(x_2, y_2, z_2) \longrightarrow (x_2 + x_t, y_2 + y_t, z_2 + z_t) \quad (3\text{-}9)$$

$$(x_3, y_3, z_3) \longrightarrow (x_3 + x_t, y_3 + y_t, z_3 + z_t) \quad (3\text{-}10)$$

So, applying t to every triangle in the model will cause every triangle to move in the desired direction and all of the triangle will maintain their size and shape.

There are two possibilities of moving the triangle either by moving the triangle itself as explained or moving the virtual world, with the triangle being the only part that does not move, in later this case is called relativity.

This is very important when we want to change viewpoints. If we were standing at the origin, looking at the triangle, then the result would appear the same in either case; however, if the origin moves, then we would move with it [6]. Figure (3-9) explains further.

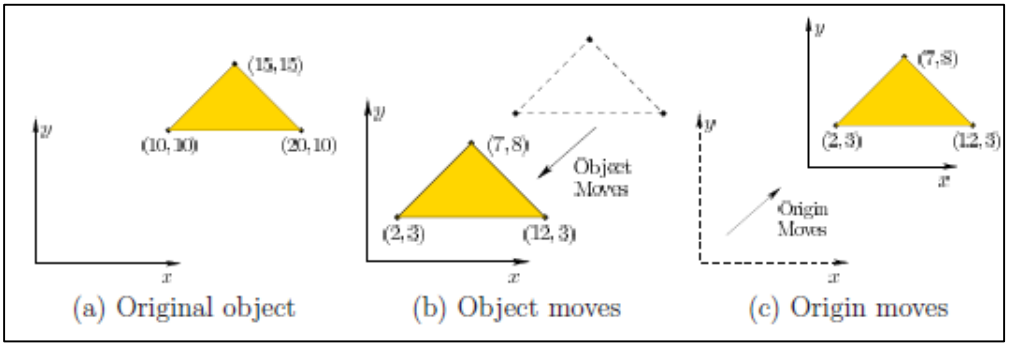


Figure 3-9: Translation and Relativity [1]

As shown in the figure there are two possible interpretations; the triangle is defined in (a). We want to translate the triangle by $x_t = -8$ and $y_t = -7$ to obtain the result in (b). If we instead wanted to hold the triangle fixed but move the origin up by 8 in the x direction and 7 in the y direction, then the coordinates of the triangle vertices change the exact same way, as shown in (c).

Beside translation and relativity, another operation is important which is rotation and we do this by changing the model's orientation in the virtual world. Rotation in 3D world is complicated than in 2D, we start by considering 2D virtual world with coordinates (x,y), Now consider a generic two-by-two matrix as follows:

$$M=\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \quad (3\text{-}11)$$

in which (m) can be a real number.

Now we multiply the matrix by the column vector (x,y)

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix} \quad (3\text{-}12)$$

In which x', y' is the transformed point

Using simple algebra yields:

$$x' = m_{11}x + m_{12}\, y \quad (3\text{-}13)$$

$$y' = m_{12}x + m_{22}\, y \quad (3\text{-}14)$$

Using notation as in equation (3-8), M is a transformation for which (x, y) → (x', y').

Now suppose we've placed two points in the plane (1,0) and (0,1), we substitute in (3-12):

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} m_{11} \\ m_{21} \end{bmatrix} \quad (3\text{-}15)$$

And:

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} m_{12} \\ m_{22} \end{bmatrix} \quad (3\text{-}16)$$

These special points simply select the column vectors on M. What does this mean? If M is applied to transform a model, then each column of M indicates precisely how each coordinate axis is changed. Figure (3-10) gives an insight of applying various matrices M to a model.

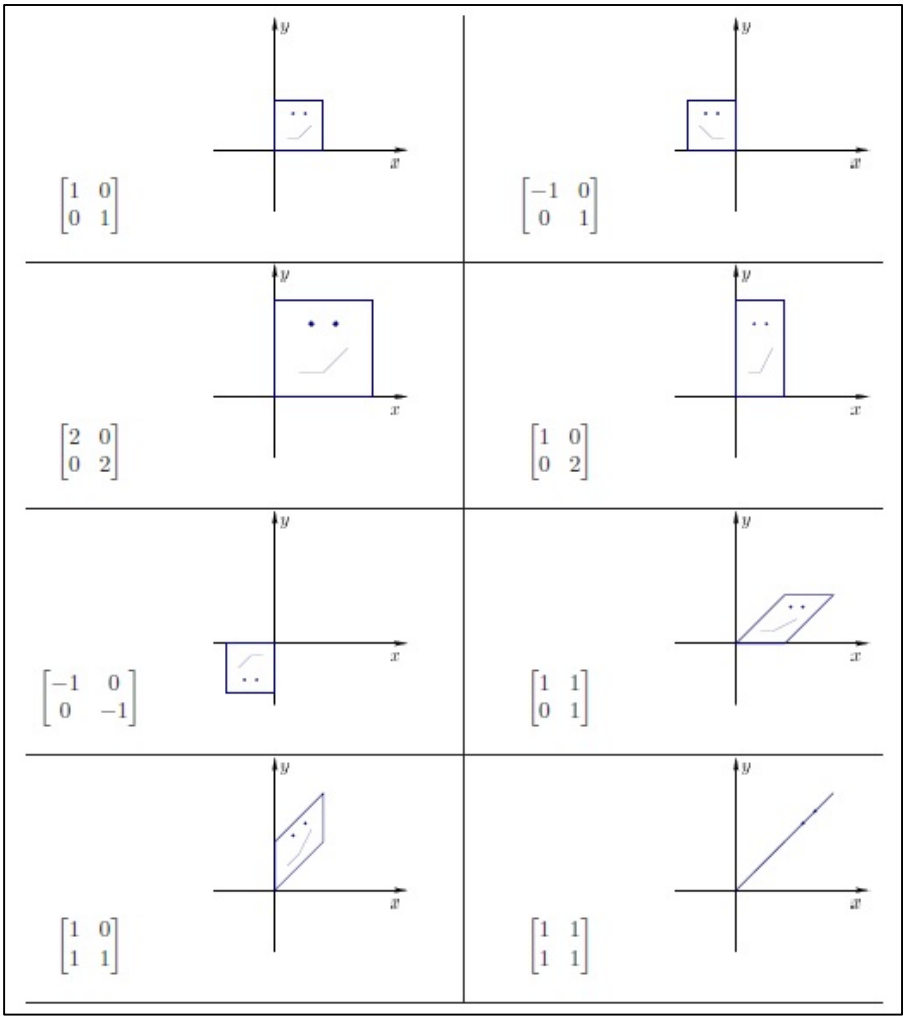


Figure 3-10: Applying Various M to a Model [1]

In the most upper left figure, the identity matrix makes no effect on the coordinates where (x,y) remains the same. In the next figure to the right shows a mirror effect where (x,y) → (-x,y). The second row from the left the coordinates are scaled by the double, where (x,y) → (2x,2y). The adjacent figure illustrates the stretch effect where the aspect ratio is distorted. On the third row the coordinated are rotated by 180 degrees. The next two figure shows the shear on the x and y axis. On the bottom right this corresponds to the case of a singular matrix.

Two of the predefined matrices produce rotation either the identity matrix which is 0 degree of rotation or the 180-degree rotation matrix. To ensure that the matrices does not distort the model M should satisfy the following rules: no stretching of axes, no shearing, and no mirror images.

To satisfy the no stretching of axes rule, the columns of M must have unit length:

$$m_{11}^2 + m_{21}^2 = 1 \; and \; m_{12}^2 + m_{22}^2 = 1 \qquad (3\text{-}17)$$

To satisfy the no shearing rule, the coordinate axes must remain perpendicular, the rule implies that the inner dot product of columns of M is zero:

$$m_{11}m_{12} + m_{21}m_{22} = 0 \qquad (3\text{-}18)$$

To satisfy the no mirror of images rule requires that the determinate of M is positive:

$$\det \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} = m_{11}m_{22} - m_{12}m_{21} = 1 \qquad (3\text{-}19)$$

The first constraint (3-17) indicates that each column must be chosen so that its components lie on a unit circle, centered at the origin. In standard planar coordinates, we commonly write the equation of this circle as $x^2 + y^2 = 1$. Recall the common parameterization of the unit circle in terms of an angle θ that ranges from 0 to 2π radians:

$$x = \cos\theta \; and \; y = \sin\theta \qquad (3\text{-}20)$$

Let $m_{11} = \cos\theta$, $m_{21} = \sin$ and substitute in equation (3-11) yielding:

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \qquad (3\text{-}21)$$

In which m_{12} and m_{22} are determined by applying equations (3-18) and (3-19), and by allowing θ to range from 0 to 2π, we can get the full range of rotation.

For the 3D case it will be consistent with the 2D, recalling the matrix form (3-5) and applying it, resulting in 9 components:

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & M_{32} & m_{33} \end{bmatrix} \quad (3\text{-}22)$$

Following the same rules for the 2D i.e. (3-17), (3-18), and (3-19), and that will result in the following:

$$m_{11}^2 + m_{21}^2 + m_{31}^2 = 0 \quad (3\text{-}23)$$

$$m_{11}m_{12} + m_{21} + m_{31}m_{32} = 0 \quad (3\text{-}24)$$

Also, the constraint det M = 1 is applied.

Yaw, pitch, and roll: again, one of the simplest ways to parameterize 3D rotations is to construct them from 2D-like transformations, as shown in the Figure. First consider a rotation about the z-axis. Let roll be a counterclockwise rotation of γ about the z-axis. The rotation matrix is given by [6]:

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3\text{-}25)$$

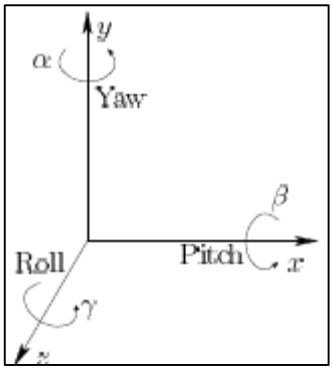


Figure 3-11: The Three-Dimensional Rotation [7]

The upper left part of the matrix looks exactly the same as the 2D rotation matrix (3-21) except the replacement of θ by γ. Similarly, let pitch be counterclockwise rotation of β about the x-axis yielding to:

$$R_x(\beta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{bmatrix} \quad (3\text{-}26)$$

Finally, let yaw be counterclockwise with respect to x and z, we'll have:

$$R_y(\alpha) = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \quad (3\text{-}27)$$

3.1.2.1.2 Rendering

Rendering starts after the completion of modelling process, with a description of how objects are arranged, materials of objects, the characteristic of light that fall into them and the place of the camera. Rendering ends with a finished image on a 2D computer monitor. There are several rendering methods stand out as producing most accurately photorealistic images.

Global illumination (shortened as GI) or indirect illumination is a general name for a group of algorithms used in 3D computer graphics that are meant to add more realistic lighting to 3D scenes. Such algorithms take into account not only the light which comes directly from a light source (direct illumination), but also subsequent cases in which light rays from the same source are reacted by other surfaces in the scene, whether reactive or not (indirect illumination) [10].

Ray tracing, beam tracing, cone tracing, path tracing, Metropolis light transport, ambient occlusion, photon mapping, and image-based lighting are

examples of algorithms used in global illumination, some of which may be used together to yield results that are not fast, but accurate. [10].

Ray Tracing: The visual attributes of each pixel in a viewport are determined by tracing a ray from a view in g position, via the pixel, into the world coordinate system. At its simplest, the pixel takes the color of whichever object is struck first by the ray. Further tracing of rays that are reflected or transmitted at the ray's intersection point with an object allows ray tracing to be used to create a large variety of optical effects [12].

These techniques simulate the behavior of lights in the real world, where a light source distributes light rays in all directions but some will reach the eye; this type of ray tracing is called forward ray tracing. Another technique is the backward tracing which traces back the light perceived by the eyes but this method in not used much in computer techniques, thus forward tracing is always referred to as ray tracing.

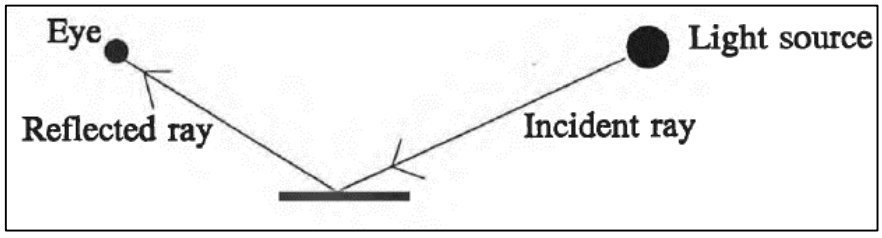


Figure 3-12: Forward Ray Tracing [12]

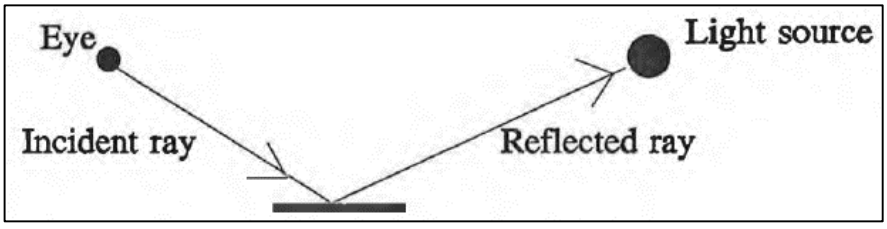


Figure 3-13: Backward Ray Tracing [12]

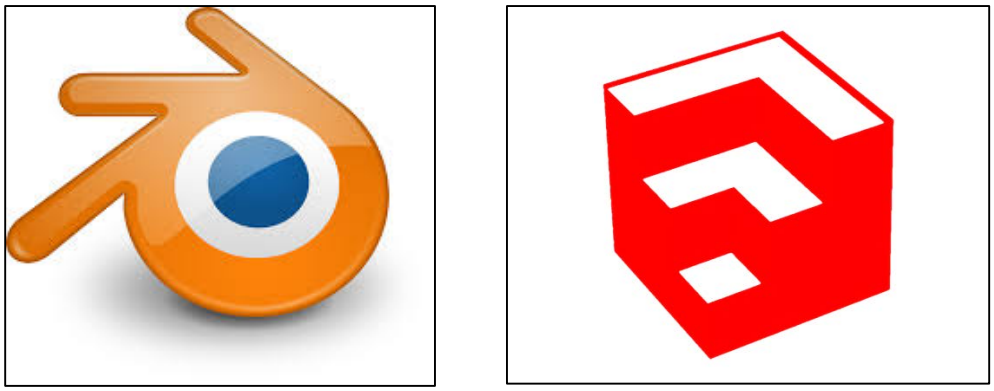
3.1.2.2 Popular 3D Modelling and Rendering Programs

3.1.2.2.1 Blender

It’s a free and open source 3D creation suite. It supports the entirety of the 3D pipeline-modeling, rigging, animation, simulation, rendering, compositing and motion tracking, even video editing and game creation. Advanced users employ Blender’s API for Python scripting to customize the application and write specialized tools; often these are included in Blender’s future releases. It's also cross-platform and runs equally well on Linux, Windows, and Macintosh computers. Its interface uses Open Graphics Library (OpenGL) to provide a consistent experience [13].

3.1.2.2.2 SketchUp

It’s a premier 3D design software from google Inc. that truly makes 3D modeling for everyone, with a simple to learn yet robust toolset that empowers you to create whatever you can imagine. In SketchUp, you can create 3D models of buildings, furniture, interiors, landscapes, and more, customize the SketchUp interface to reflect the way you work, share 3D models as walkthrough animations, scenes, or printouts, with realistic light and shadows. [14].



(a) (b)

Figure 3-14: (a) Blender [13] and (b) Google SketchUp [14]

3.1.2.3 Popular Game Engines

3.1.2.3.1 Unreal Engine

Unreal Engine (UE), initially released on 1998 by Epic Games, is a complete suite of game development tools, powering hundreds of games, simulations and visualizations. It is one of the most advanced engines to date, delivering top quality visuals while providing users with a large variety of tools to work with everything they need.

Due to its capabilities, efficient design and ease of use it is well-appreciated engine from hobbyists to development studios. It is also available for free. Developers can also port their projects to mobile devices, both iOS and Android. Unreal Engine also works with Virtual Reality. Finally, UE also gives access to its users with to a marketplace, to buy re-usable content and add to their project, speeding the development process [15].

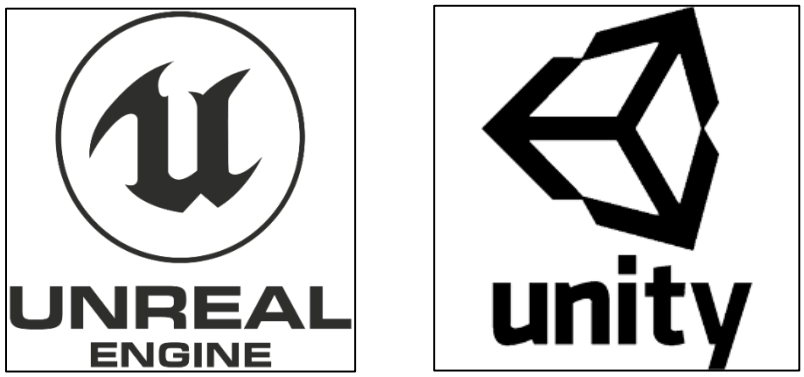
3.1.2.3.2 Unity 3D

Initially released on 2005, is a flexible and powerful development platform for creating high quality 2D and 3D games. Emphasizing on portability, Unity currently supports over 20 platforms, including PCs, consoles, mobile devices (iOS and Android) and websites.

Additionally, many settings can be configured for each platform. As a result, Unity can detect the best variant of graphic settings for the hardware or platform the game is running, thus optimizing performance and sacrificing visual quality if necessary.

Apart from its next-generation graphical capabilities, Unity also comes with an integrated physics engine (Nvidia PhysX). Much like Unreal Engine, Unity offers developers an Asset Store to buy reusable content and assets for

use in their project. To sum up, due to its ability to efficiently target multiple platform at once and user-friendly environment, this game engine is an ideal choice for a large portion of developers [16].



(a) (b)

Figure 3-15: (a) Unreal Engine [15] and (b) Unity 3D [16]

3.1.2.3.3 FlightGear

It is an open-source flight simulator released on April 1996. It supports a variety of popular platforms (Windows, Mac, Linux, etc.) and is developed by skilled volunteers from around the world. The goal of the FlightGear project is to create a sophisticated and open flight simulator framework for use in research or academic environments, pilot training, as an industry engineering tool, for Do It Yourself (DIY-ers) to pursue their favorite interesting flight simulation idea, and last but certainly not least as a fun, realistic, and challenging desktop flight simulator [17].

3.1.2.3.4 X-Plane:

The world's most comprehensive and powerful flight simulator for personal computers, and it offers the most realistic flight model available. X-Plane is not a game, but an engineering tool that can be used to predict the flying qualities of fixed- and rotary-wing aircraft with incredible accuracy.

Because X-Plane predicts the performance and handling of almost any aircraft, it is a great tool for pilots to keep up their currency in a simulator that flies like the real plane, for engineers to predict how a new airplane will fly, and for aviation enthusiasts to explore the world of aircraft flight dynamics. X-Plane is used by world-leading defense contractors, air forces, aircraft manufacturers, and even space agencies for applications ranging from flight training to concept design and flight testing [18].

One simulator that uses Xplane as its visual system is the Antonov AN-26B simulator based in Khartoum airbase, upon which Xplane is used for the entire visual system as well as to control weather conditions, daytime, date of flying, and everything else related to the visual scene.

(a) (b)

Figure 3-16: (a) Antonov AN-26B Simulator Using (b) Xplane Visual Engine

3.1.2.3.5 Prepare3D

Pronounced as "Prepared" or shortened as P3D is a visual simulation platform that allows users to create training scenarios across aviation, maritime and ground domains. Prepar3D engages users in immersive training through realistic environments. Ideal for commercial, academic, professional, or military instruction. Prepar3D can be used to quickly create

learning scenarios anywhere in the virtual world, from under water to sub orbital space [19].

Prepar3D can be used for wide range of learning scenarios including vehicle procedures training, cockpit familiarization, flight planning, air traffic controller training and emergency response preparation. As a commercial-off-the-shelf product, Prepar3D provides a cost-effective training platform that evolves with technology [19].

Same as Xplane, prepar3D is also utilized in real flight simulators for creating visual scenes for both civilian and military applications; one example of the military application of P3D is the usage of this engine in the visual system of a jet trainer K8, which is an advanced jet trainer-fighter airplane. Figure (3-17) shows a visual of the runway marked 04 using P3D with three-channels projection scheme and Mylar screen, each channel is dedicated to one projector and all together overlap to give the full picture.

(a) (b)

Figure 3-17: Prepar3D in (a) Military and (b) Civil Applications

On the other side, for civilian utilization of P3D is in a generic flight simulator for Beechcraft Super King Air B200; in which the airplane is a twin

turboprop engine used in short distance travel carrying 7-9 passengers, P3D is used as a single channel connected to a display to give the visual perception.

Other dedicated game engines are for specific simulators such as VirualF as shown in the figure below for the helicopter AK1-3, and as depicted in figure (3-18) the simulator uses 3-channels to display the visual scene and for every channel there is a corresponding computer having the engine installed.

Figure 3-18: A 3-Channels Dedicated Visual Engine

3.2 Classifications of VR Systems:

The sensory perception impression delivered to the human determines the level of immersion and classification of a VR system**,** in ideal system

information should be presented to all human senses, but in practice this somehow different because not all human senses are targeted. So we can classify the level of immersion into these categories.

3.2.1 Desktop VR

It's considered as non-immersive system and it's the least implication of VR. Sometimes it's called Window on World (WoW), usually it uses conventional display as an output.

3.2.2 Fish Tank VR

This system is more complex than Desktop and it supports head-tracking and conventional display.

3.2.3 Immersive Systems

The ultimate version of VR systems. They let the user totally immerse in computer generated world with the help of Head Mounted Display (HMD) that supports a stereoscopic view of the scene accordingly to the user's position and orientation. These systems may be enhanced by audio, haptic and sensory interfaces [9].

3.3 Basic Concept of VR:

Figure (3-19) shows a concise view of the main principles of a typical VR system; where an input that might be mouse, keyboard, game controller, head tracking …. etc., sends all the relevant data to the computational unit namely the VWG that creates another world which could be completely synthetic or recorded physical world. This unit then outputs the appropriate views to displays. The primary renderer for VR is the visual rendering since vision contribute up to 70% of the human senses followed by hearing which has a percentage of 20% contribution; this means that human vision provides

most of the information to our minds. Other human senses contribution is as follows:

Smell	5%
Touch	4%
Taste	1%

Table 3-1: Contribution of Human Senses

Touch in general is not that significant unless the application requires touch so, smell and taste are not considered yet in VR because the relative difficulty in implementation.

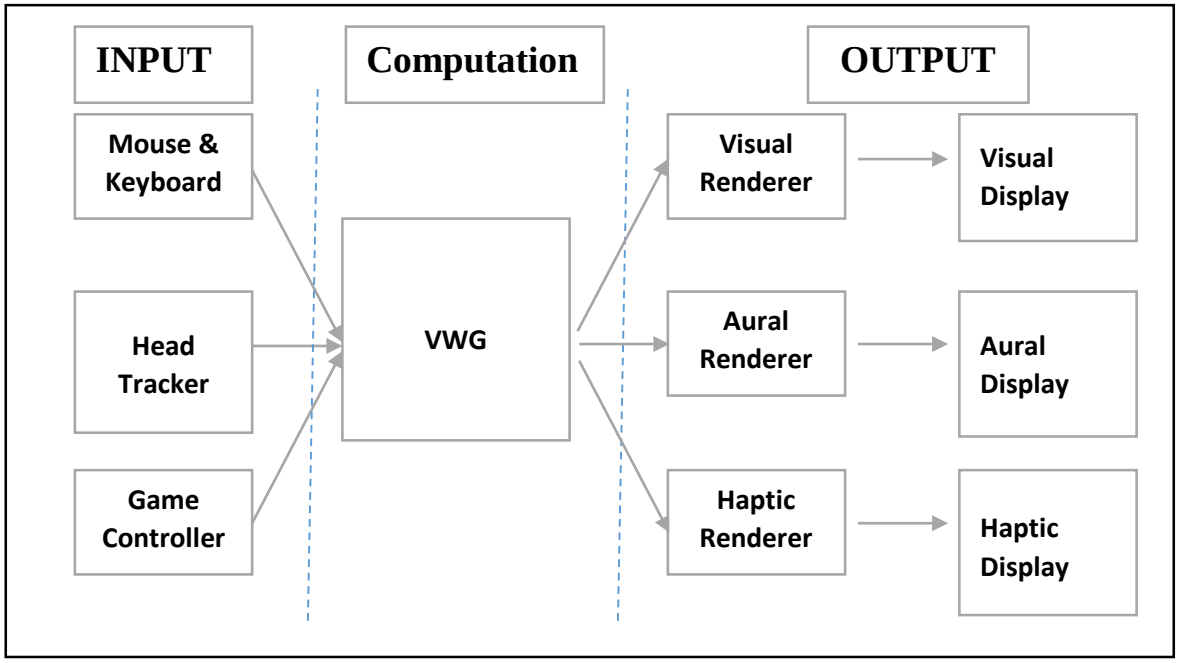


Figure 3-19: Schematic of a Typical VR System

3.4 Human Factors:

Virtual environments are meant to fool the sense of the human being and plunge the user in the VR world, this task is not that simple and it is very complicated because the user must feel immersed in the environment and in return this solution should be feasible somehow. So, we have to make deeper

understanding of human factors and synchronization of stimuli with user's actions.

3.4.1 Visual Perception

3.4.1.1 Field of View (FOV)

Human horizontal field of view is 150°: 60° towards the nose and 90° to the side, while the vertical FOV is 180° of total horizontal viewing range with a 120°. Figure (3-20) shows the horizontal and vertical FOV of human vision.

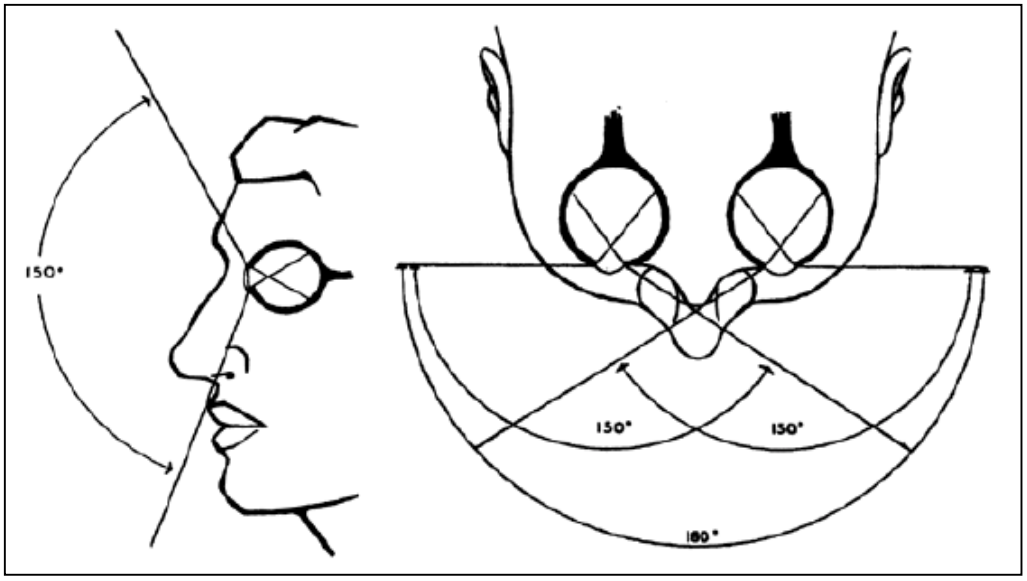


Figure 3-20: Human Vision FOV [9]

3.4.1.2 Visual Acuity

It is the sharpness or clarity of vision, measured by the ability to distinguish between letters and numbers at a certain distance according to fixed standard. A famous method to determine the visual acuity is the Snellen's chart as in Figure 3-21.

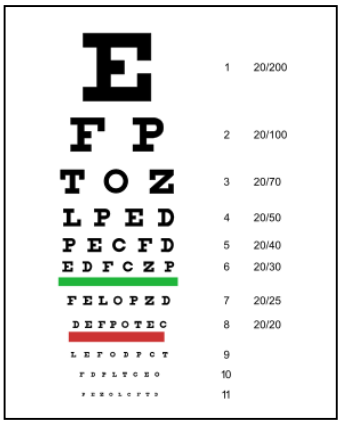


Figure 3-21: Snellen's Chart [9]

3.4.1.3 Temporal Resolution

Temporal resolution of the eye refers to the flickering phenomena perceived by humans, when watching a screen e.g., CRT (Cathode Ray Tube) that is updated by repeated impulses. Too low refresh rates, especially for higher luminance and big displays, causes the perception of flickering [6].

3.4.1.4 Luminance and Color

The human eye has a dynamic range of ten orders of magnitude which is far more than any current available display can support. Therefore, special color mapping techniques must be used to achieve possibly the best picture quality [9].

3.4.1.5 Depth Perception

To generate depth information and stereoscopic images the brain extracts information from the pictures the eyes see and from the actual state of the eyes. This bits of information are called depth cues. All of the depth cues may be divided into two groups: physiological (like accommodation, convergence or stereopsis) and psychological (like overlap, object size, motion parallax, linear perspective, texture gradient or height in visual field) [20].

3.4.2 Human Physiology

Our bodies were not designed for VR, by applying artificial stimulation to the senses; we are disrupting the operation of biological mechanisms that have taken hundreds of millions of years to evolve in a natural environment.

We sometimes provide inputs to our brains that might not be consistent with our pervious experiences, our bodies may adapt with the new stimuli leaving us unaware of the flaws in the VR system.

Furthermore, we may develop greater awareness or ability to understand these 3D scenes which were hard or ambiguous to understand before, but unfortunately this will lead to increased fatigue or headache; the reason for this is that our brain exerts more efforts to interpret the stimuli. The worst case is the onset of VR sickness, which typically involves symptoms of dizziness and nausea [9].

3.5 Current VR Applications

3.5.1 Video Games

Being a part of a video game or entering a video game was a dream for many people and has been around for over decades, luckily this dream came true with the introduction of VR. And excellent example of this is the interactive Pokémon Go game; where the player uses his smart phone camera as see through the outside scenes and tries to catch Pokémons this in term a good example of Augmented Reality AR. Figure (3-23) shows more about the idea behind this game.

Figure 3-22: Pokémon Go Video Game [1]

3.5.2 Immersive Cinema

Feeling a part of the movie is this what looks like in immersive cinema, where all the possible realization is put forward; the use of motion seats that can move according to a specific motion or vibrate.

3.5.3 Telepresence and Teleoperating

It is a specific kind of virtual reality that simulates a real but remote (in terms of distance or scale) environment. Another more precise definition says that telepresence occurs when at the work site, the anipulators have the dexterity to allow the operator to perform normal human functions; at the control station, the operator receives sufficient quantity and quality of sensory feedback to provide a feeling of actual presence at the worksite [9]. The idea of teleoperating is shown in figure (3-23).

3.5.4 Virtual Societies

Virtual reality is also capable to connect people through societies in terms of avatars connected to real people. Those people could gather together for various reasons such as common interests, educational purposes, or simply escaping from real life.

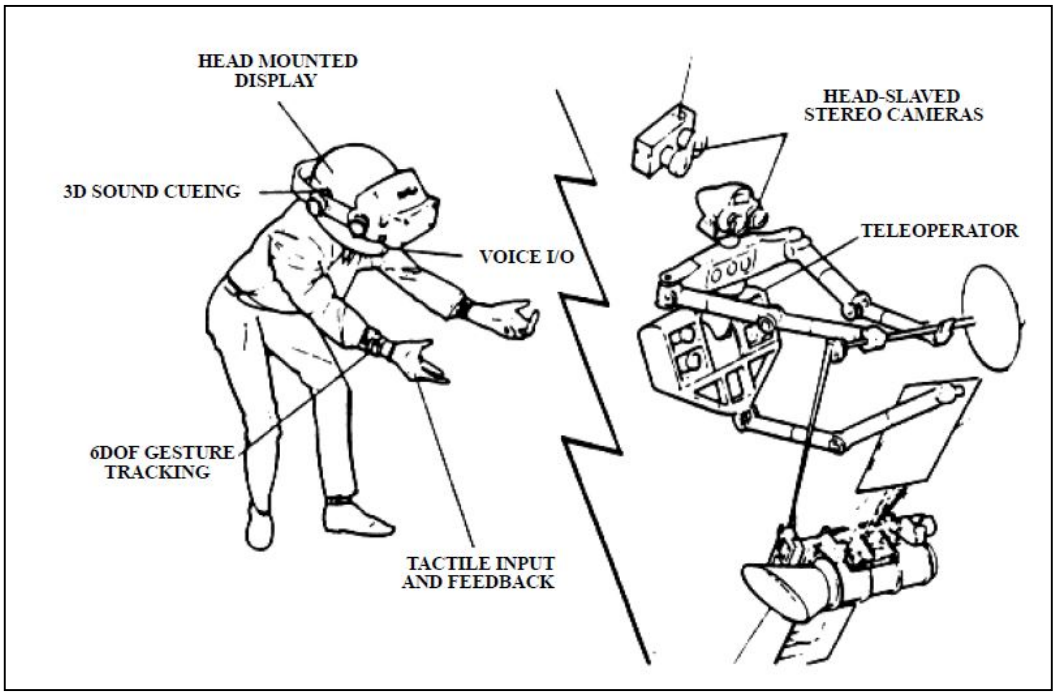


Figure 3-23: The Idea of Teleoperating [9]

3.5.5 Training and Education

Using VR and first-person perspective education might also be achieved in different aspects for example engineering concepts and theories, visualizing the movement of air in wind tunnels. One of the most common examples is the flight simulation and that what we would like to achieve through this research. Figure (3-24) shows an example of a flight simulator used by United States Air Force (USAF) where the pilot sets inside the cockpit of the simulator and wherever the pilot looks, he is surrounded by the projected scenes this system is widely recognized in VR as CAVE.

3.5.6 VR Headsets

The common trend in nowadays technology is the portability and mobility starting from the cinemas and TV, so instead of going to the cinema to watch a movie with a high resolution film and large projection screen, people prefer to watch it at home with their families or sometimes even using their own handheld smart phones, so following this trend also in VR headset where also used with smart phone and personalize preferences of mobility rather than using the immersive cinema as mentioned before.

Figure 3-24: A Flight Simulator Used by USAF Utilizing CAVE Concept [1]

3.6 Virtual Reality Flight Simulator

A flight simulator is a device that recreates an aircraft and its environment or any events where it flies. It is a world where someone who wants to become a pilot, or someone who just wants to learn how to fly a plane, or just to play without having to use the original plane. In this case, the flight simulator is very important for the learning of prospective pilots or

airline pilots. To support this pilot prospective study, the flight simulator should be made as closely as possible to the real situation they would have when riding a plane. In this manner, this device needs to include equations of how an aircraft is flying, how flight controls react when it is triggered, effects of other aircraft systems, and reaction of aircraft to external factors such as damping, gravity, air density, turbulence, etc [8].

The hardware of the VRFS varies as the purpose varies whether it is generic or specific; generic flight simulator represents a family of airplanes or similar airplanes in terms of common systems, common switches and indicators, or sometimes limitations, so the simulator can be used to serve more than one type of aircraft and vice-versa for the specific simulator that serves only one type.

Figure 3-25 shows a possible configuration for data flow through the hardware components of VRFS. The system is distributed as the modules are generally located on different computing environments due to requirements of high computing power. Also, network connected computers are needed where multiple users are involved. The communication among the modules is provided by Local Area Network (LAN). The data flow through the modules would be different where more users are involved or additional modules are executed [10].

Each module can be located on a different workstation where high performance is needed; however, end-to-end latency of the distributed system increases with this configuration due to network latency.

Any flight simulator has three principal tasks: image display, image generation, and flight dynamics. Many flight simulators use either domes or

CRTs for out-the-window image display. However, military simulators have used see-through head-mounted displays (HMDs) for some to display the out-the-window imagery with physical gauges and dials still visible [10].

Any flight simulator has three principal tasks; image display, image generation, and flight dynamics. Many flight simulators use either domes or CRTs for out-the-window image display. However, military simulators have used see-through head-mounted displays (HMDs) for some to display the out-the-window imagery with physical gauges and dials still visible [9].

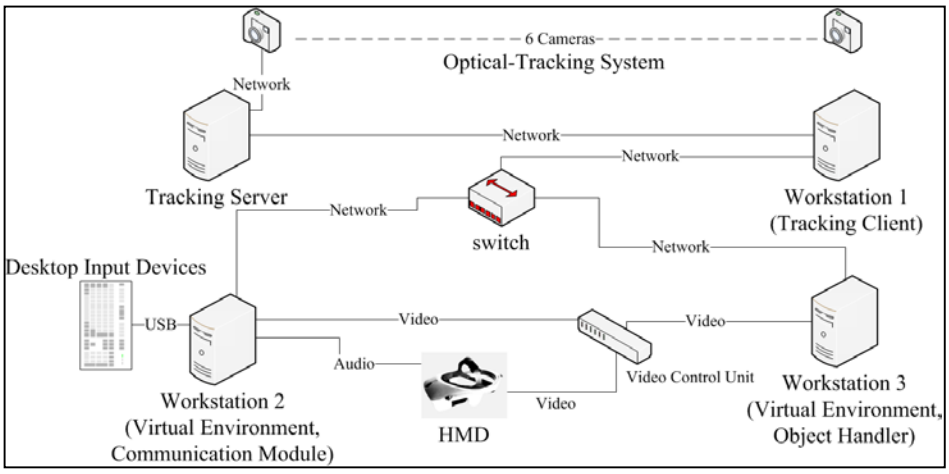


Figure 3-25: Possible Configuration for Data Flow of a VRFS [7]

CHAPTER FOUR

VIRTUAL REALITY FLIGHT SIMULATOR

CHAPTER FOUR

VIRTUAL REALITY FLIGHT SIMULATOR

4.1 3D Modelling

The block diagram in figure (4-1) shows the overall concept of the modeling which consists of: the plane being modeled an Embraer Regional Jet (ERJ-145), the plane maker application for creating the model. The model is then placed in xplane engine for testing, after testing the model in xplane then the model is transferred to blender for creating a detailed replica of cockpit of the aircraft and afterwards the cockpit is integrated with the model and tested back in xplane engine.

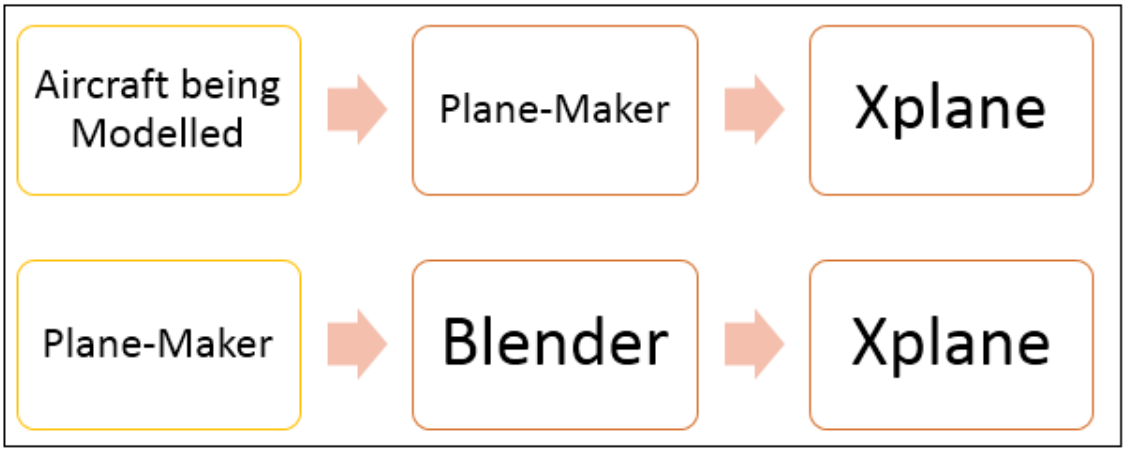


Figure 4-1: Block Diagram of the Model

The aircraft is ERJ-145 (Embraer Regional Jet) which is a Brazilian made aircraft with seating for a total of 50 passengers is specifically selected because its data package is widely available from different resources, as well as the aircraft is somehow averagely complexed in regards with other types of airplanes such as Airbuses or Boings.

Plane-Maker is a program bundled with X-Plane that lets users design their own aircraft. Using this software, nearly any aircraft imaginable can be built. Once all the physical specifications of the airplane have been entered (e.g., weight, wing span, control deflections, engine power, airfoil sections, etc.), the X-Plane simulator will predict how that plane will fly in the real world using the blade element theory; airplanes are saved in Plane Maker as an .acf file format or extension, these files are then opened and flown in the X-Plane simulator.

As mentioned in figure (4-1), the first point to start with is plane-maker application, so as seen in the next figure (4-2) the fuselage of the plane must be created by using the ERJ-145, the side view, top and bottom view.

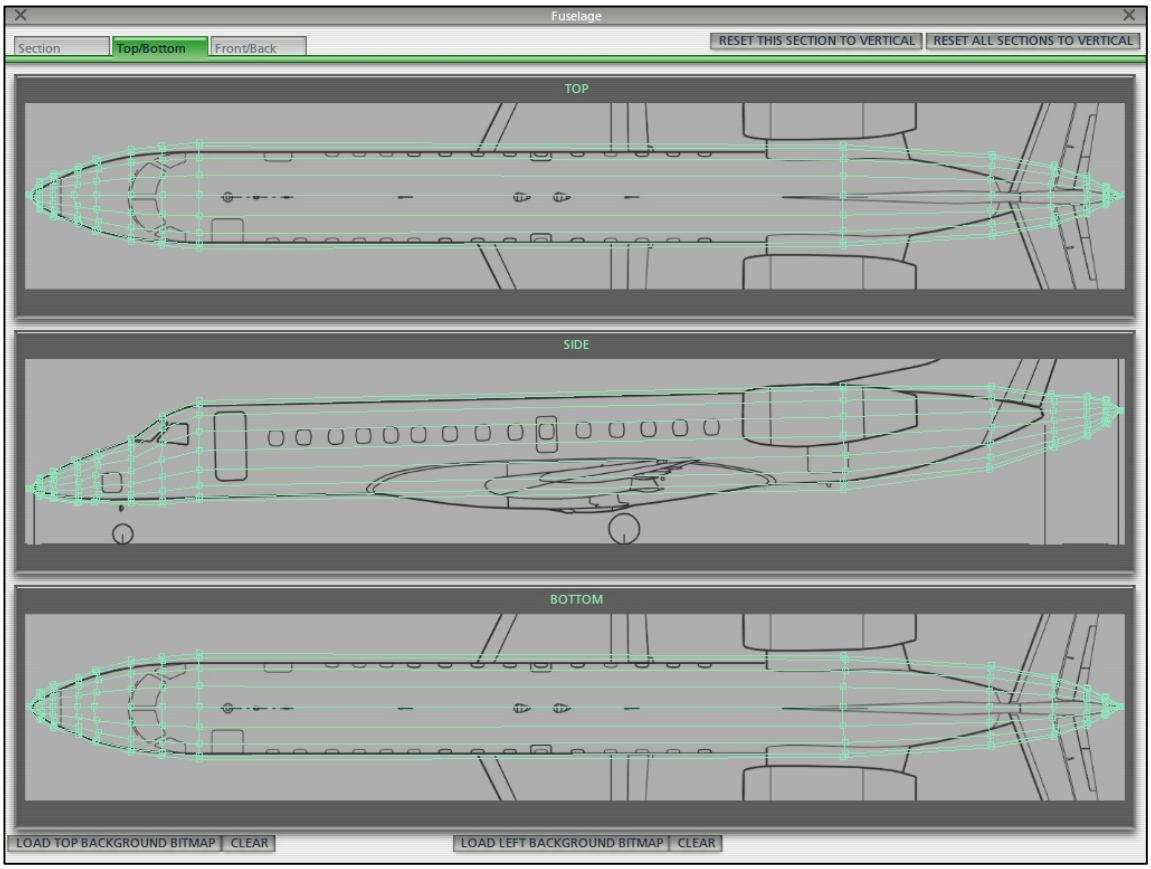


Figure 4-2: Modelling the Fuselage in Plane-Maker

As shown in the previous, the cross-section of the plane is laid down and the corresponding lines are dragged to be aligned with the cross-sections. The top and bottom views are mirrored meaning any alteration in one will affect the other. The side view is merely a duplicate for both sides since they are exactly the same.

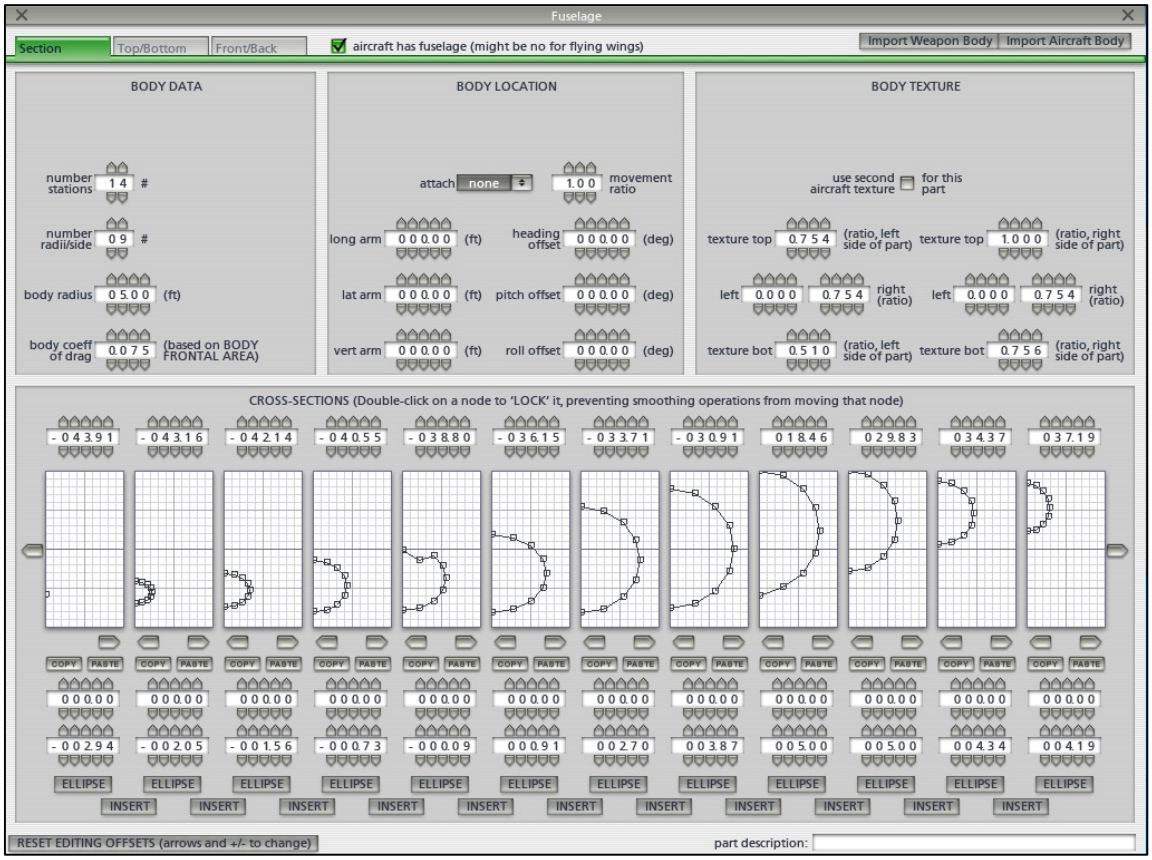
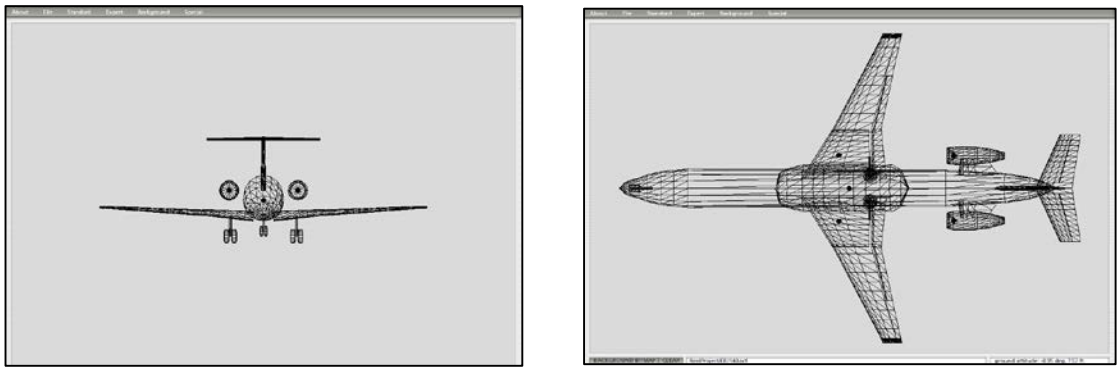


Figure 4-3: Fuselage Data, Body Location, And Body Texture of the Aircraft

In Figure (4-3) the fuselage data is entered in terms of multiple sections, and cross sections. during the creation of the fuselage and dragging lines to cover the fuselage geometry, sometimes the fuselage must be smoothed and this is done by clicking over the ellipse tap to make the fuselage in a way that corresponds to the aerodynamics.

Adding the other components of the aircraft, i.e. wings, ailerons, flaps, other control surfaces, rudder, elevator, engines, engines parameters and specifications, engines nacelles, pylons, and landing gear will result in a complete aircraft as shown in figure (4-4). According to the program specification the model is scaled down to 1:48.



(a) (b)

Figure 4-4: Front (a) and (b) Top View of ERJ-145 in Plane Maker

In designing this VR flight simulator, only the point of interested is modelling the cockpit with its related instruments, but t in either way model the whole aircraft since all the parameters are needed for the aircraft to make it fly in X-plane and to make all the instruments alive in the cockpit.

One thing to note in modelling this aircraft is that the modelling uses the polygonal mesh as discussed in chapter 3, also other components of the cockpit e.g. instruments, switches, …etc., will be modelled using polygonal meshes.

4.1.1 3D Modelling in Blender

The next step is to model the cockpit in a more complicated yet meticulous program which is Blender, here Blender version 2.48 is used although up to version 2.80 is released, but older version is selected because

it's less complicated and easy to use. There must be a python 2.5.X compiler in order for blender to run.

4.1.1.1 Understanding Normals

Normals, as applied to 3D modeling, describe the way that a polygon's visible surface is facing. Within Blender's default settings, polygons are not two-sided objects and, in fact, are only visible from one side [21].

In this model most of the normals will be facing outside unless where some places need the contrary. For example, when working on the inside panels of the cockpit the normals will face the outside so when we sit inside the cockpit all instruments and other components will be visible and vice versa for the outer skin of the cockpit where the vertices' normals will be facing inwards.

4.1.1.2 3D Object Construction

In the computer graphics industry, the points of a model are called vertices. The lines between vertices are called edges. The three-dimensional forms that are created when at least three vertices are connected by edges are called polygons. If a polygon has a surface on it, this is called a face.

When polygons or faces are arranged in such a way that they create a three-dimensional form, this is called a mesh. Some meshes of basic 3D forms are so widely used that they come premade in 3D programs. The objects in figure (4-5) are some of the basic mesh forms available in Blender.

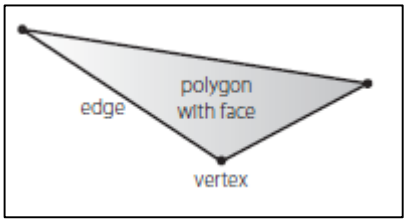


Figure 4-5: Edge, vertex, polygon, and Face Concepts

4.1.1.3 3D Cockpit

By creating a mockup of the cockpit resembles the basis or reference for the other parts of the cockpit. Afterwards the front panel of the cockpit will be modeled and following other parts of the cockpit i.e yoke, pedals, central pedestal control panel, and finally the overhead panel.

Figure 4-6 illustrates the process of creating the outer parts of the cockpit. The windshield is created first in regards to the dimensions of the aircraft. Then the windshield acts as the base for modelling the sectioned-part of the cockpit.

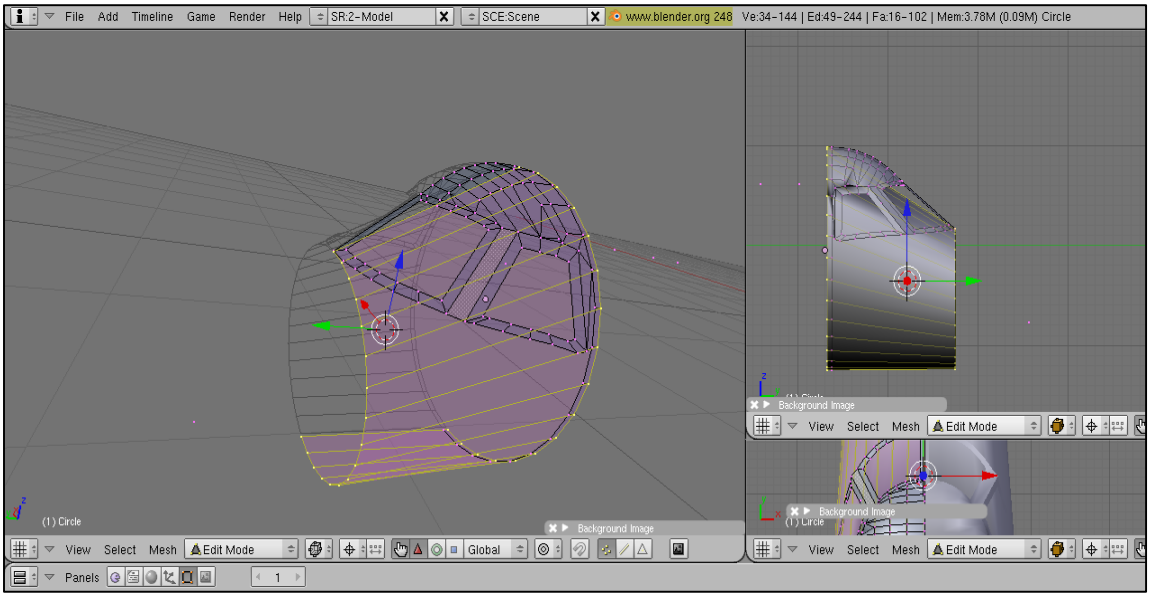


Figure 4-6: The Process of Modelling the Outer Skin of the ERJ Cockpit

The final result of this work is apparent in figure (4-7). In which some blending techniques are used such as mirror modifier; since we can divide the cockpit latterly by two halves, we can only create one half only and then add the mirror modifier in blender to reflect the other half.

It's visible the use of meshes in creating the outer part of the cockpit, rectangular meshes are used with faces to link or join all meshes together. All

faces are editable in terms of rotation around all axis as well as vertices and edges.

The windshield is left empty since it resembles the real windshield in the aircraft and it acts as the view point when the cockpit is placed in the engine, hence it will be left blank with no alteration.

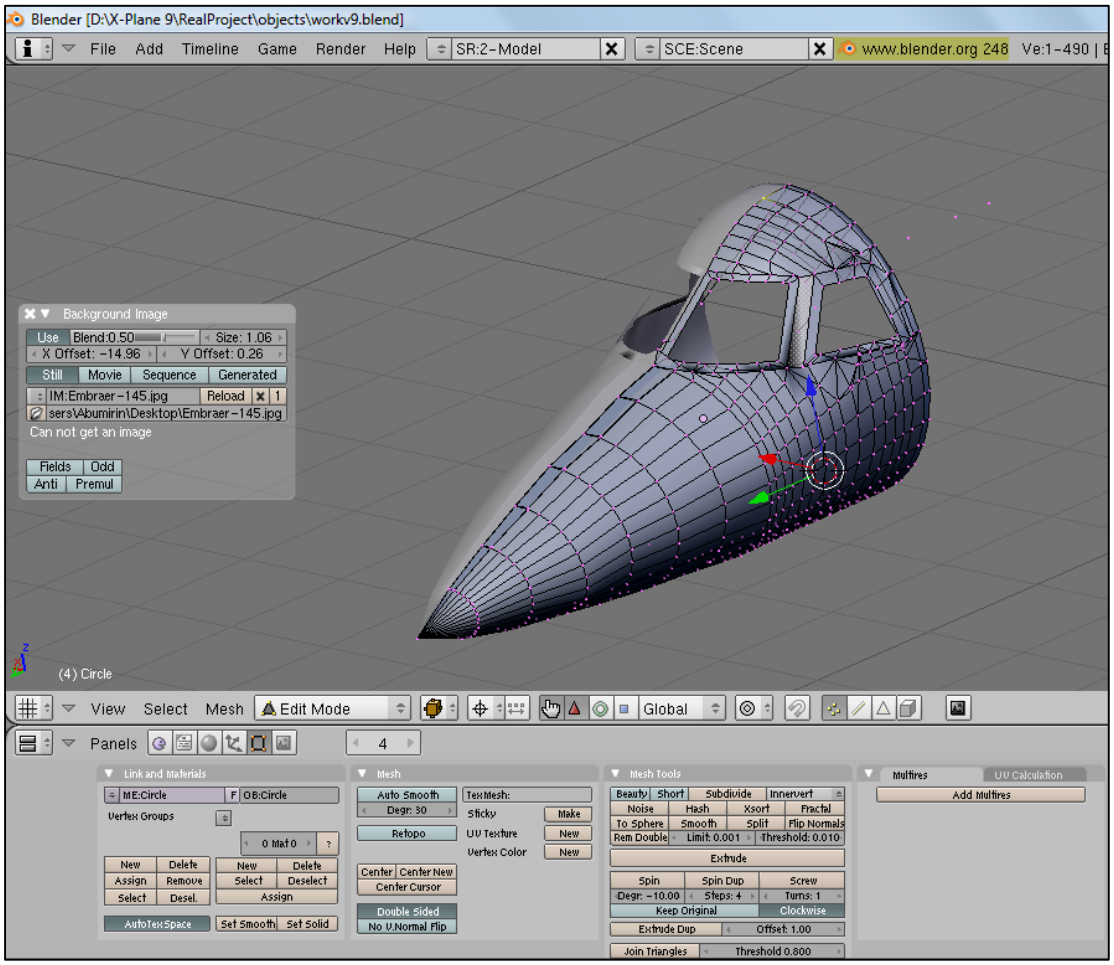


Figure 4-7: The ERJ Outer Skin of the Cockpit

The next step is the creation of the inside panels of the cockpit, namely the center main instrument and glare shield panels which contains the following instruments: The Primary Flight Display (PFD), which presents information about primary flight instruments, navigation instruments, and the status of the flight in one integrated display.

The other significant display is the Multi-Function Display (MFD) which acts as a backup for the PFD can be used to display anything, but usually it's used for traffic display, route selection and weather and terrain avoidance.

The third display is the Engine Indication and Crew Altering System (EICAS), as the name implies it gives all relevant data concerning the engines from fuel flow up to engine compression ratio. Besides the engines data it also gives crew annunciations, and remedial actions known as checklists.

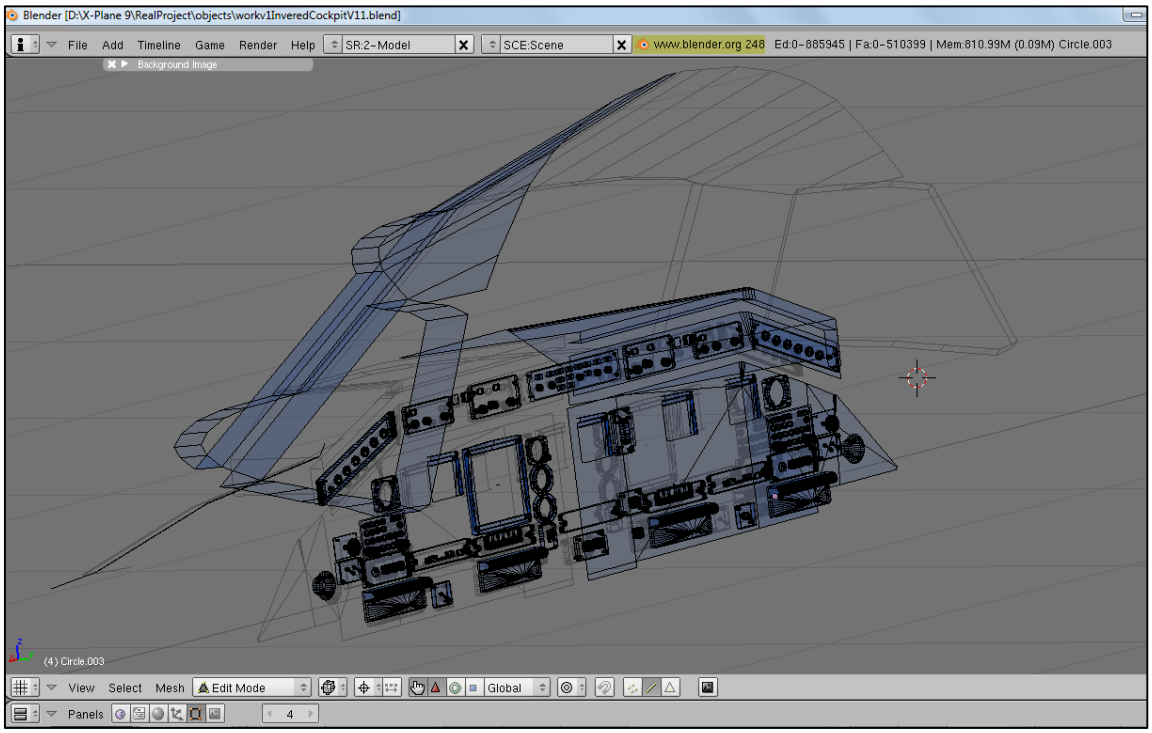


Figure 4-8: The Instruments and Glareshield Panels

The main instruments panel contains also three standby conventional indicators for usage in case the MFD and PFD malfunction or any other emergency state, these instruments are the attitude indicator, airspeed indicator, and altimeter.

In figure (4-8) the instruments and glare shield panels are modelled as well as the other switches, buttons, and knobs. The reference for drawing the panels is a 3D view from images of the cockpit taken from different angles all combined together.

Creating the panels separately and then combing them back together is quite useful, and this method provides that all previous work on panel is kept isolated and only alterations are made to the current panel being created. In figure (4-9) the overhead panel is constructed using this method.

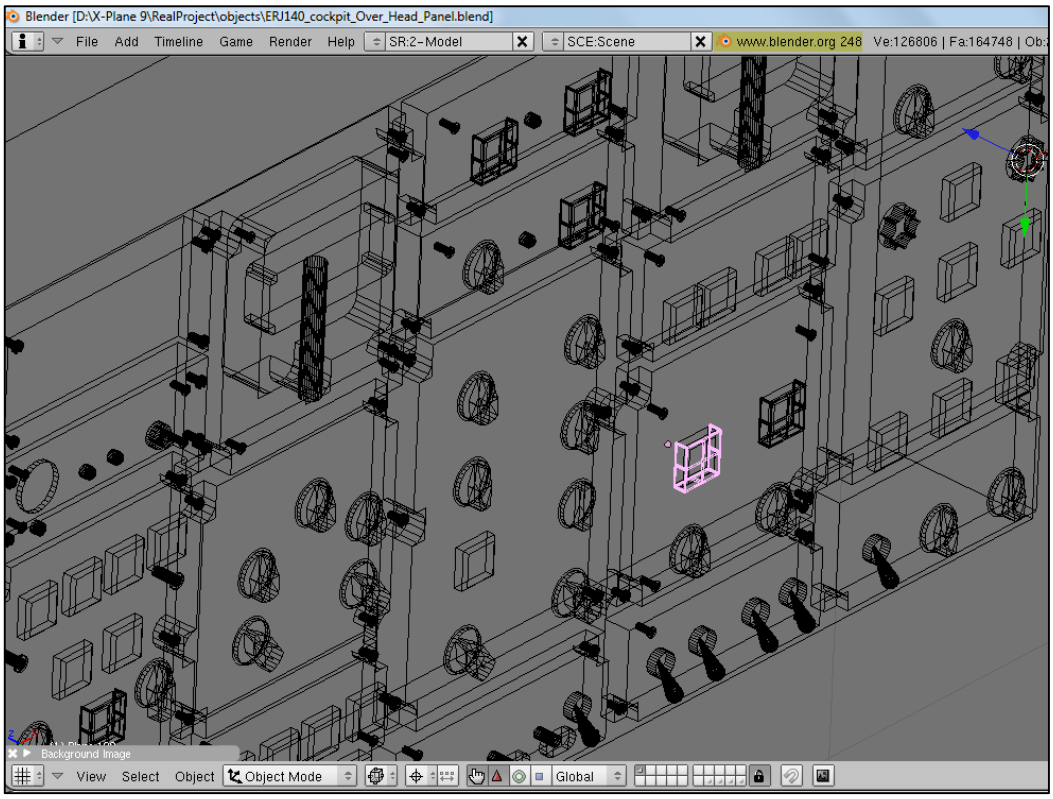


Figure 4-9: The Overhead Panel

The next figure is the final version of the cockpit comprising of all panels. All of the indicators, switches, controls, do have the same colour along with the transparent or semi-hollow indicators, to make things work, it's needed to integrate the indicators panels with that one of Plane-Maker using the panel region handler from the plug-in xplane2blender.

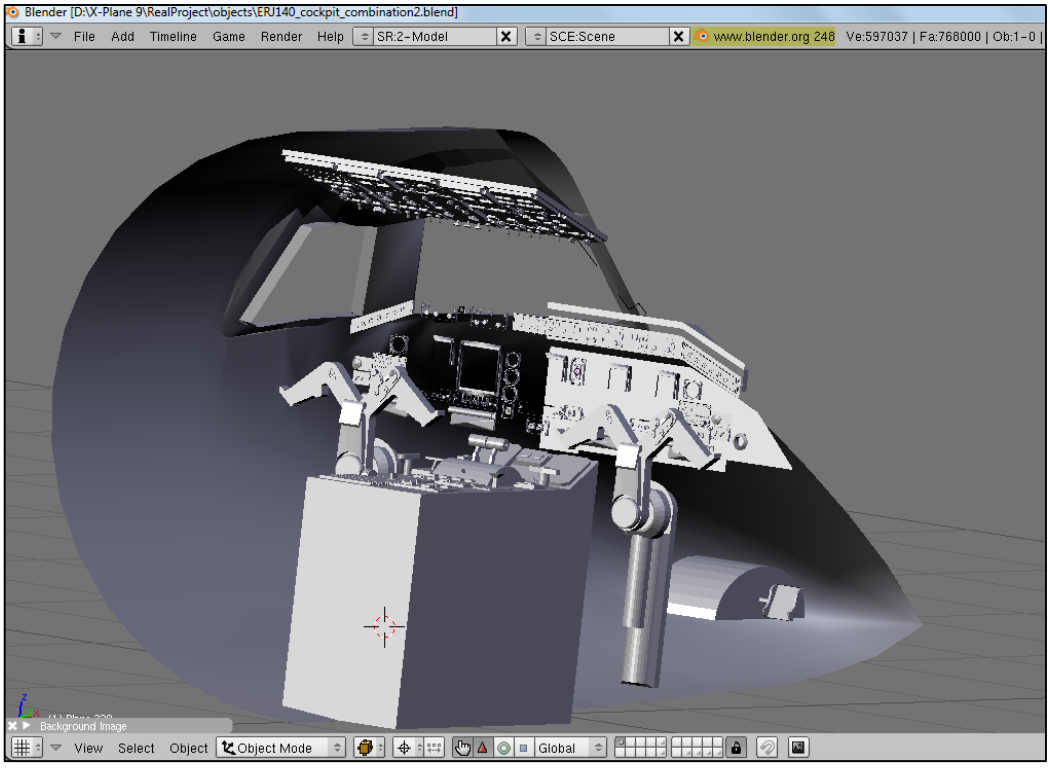


Figure 4-10: The 3D Model of the ERJ-145 Cockpit

4.2 Tweaking the Cockpit

4.2.1 Panel Region

The idea of the panel region is using a texture in the 3D cockpit, this in return allows to have moving indicator or glass display in the 3D cockpit using only 2D texture. In order to cast the image into the 3D model we have to do this in the UV/Image Editor in Blender.

The panel region is a sub-area of the panel, and it must be a power of 2 with dimensions up to 2048x2048 pixels, it also features 4 different regions and can set to be overlapped. In this perspective we packed all required panels in only one panel to make it easier to be processed.

In figure (4-11) shows the steps necessary to skin the parts of the cockpits. In the figure we used two views in Blender the 3D and UV/Image

Editor view; so, we have to select the areas where we want to skin.

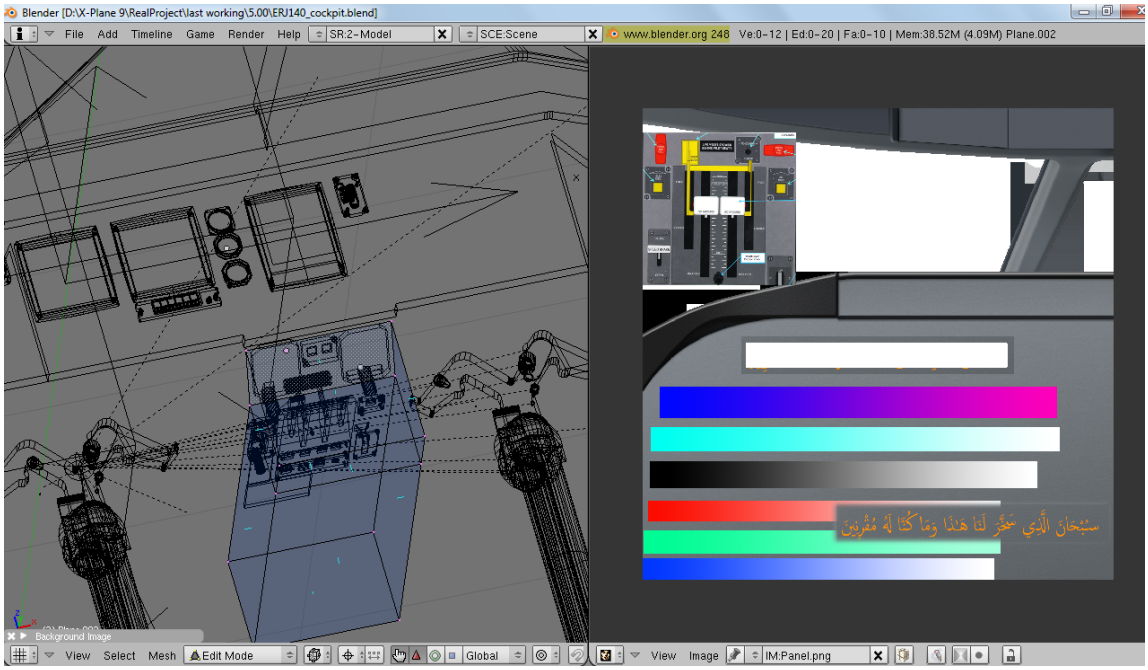


Figure 4-11: UV Mapping

Here the standby Airspeed indicator is skinned using the command project from view and then determining the right place for the indicator in the UV/Image editor. One thing to note that, we've created 3D panel already in Plane-Maker containing the essential indicators.

The same concept applies for all other indicators as well as the PFD, MFD and EICAS. The major constrain and sometimes this constrain prevents a successful UV mapping which is that the modeled part must have face or faces to assign the texture to.

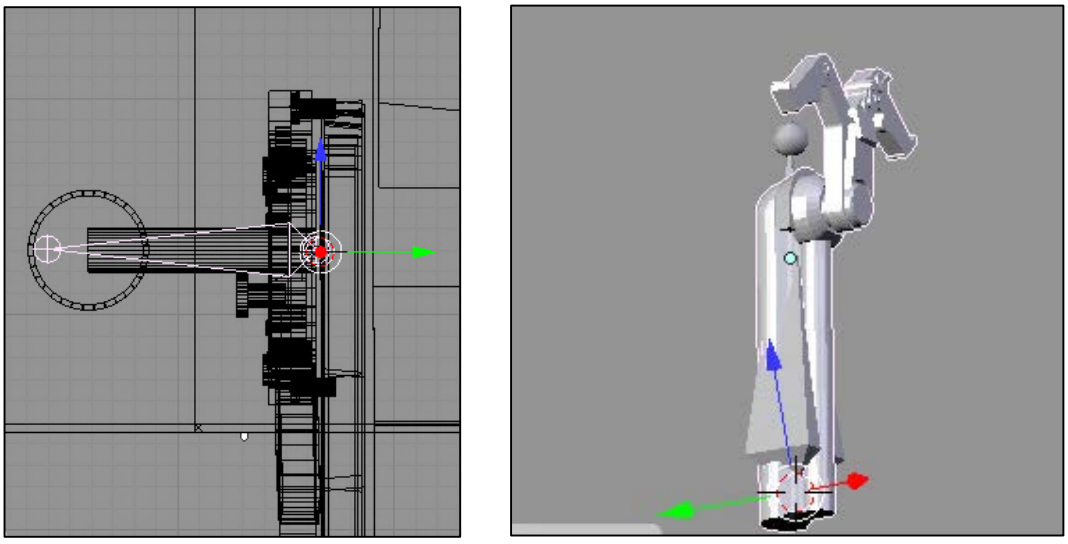
4.2.2 Datarefs:

A dataref is a single bit of published information. dataref key frames describe how X-Plane should play the animation. This is different from traditional animation, where the animation tells a rendering engine (such as a 3D movie exporter) how to move geometry over time. Based on the value

of a dataref inside X-Plane, X-Plane's rendering engine will interpolate between keyframes and display your model at specified positions.

For instance, taking the landing gear handle, it must be first assign an armature to the handle to make it one part. An “armature” is a type of object used for rigging. A rig is the controls and strings that move a marionette (puppet). Armature object borrows many ideas from real-world skeletons, and just like a real skeleton an Armature can consist of many bones [22].

So back to the landing gear handle, now to animate or make the handle clickable and it reacts when it is up consequently the landing gear goes up and vice versa, an armature is parented to the handle and having two frames for the handle up and down.



(a) (b)

Figure 4-12: Assigning an Armature to the Gear Handle (a) and (b) the Yoke

In figure 4-12, the armature with pink colour, is assigned and parented merely for the gear handle. Other moving parts or elements of the cockpit are assigned different armatures with different datarefs. The movement of the handle is pivoted at the cursor and only two frames are needed since the

position of the gear is either up or down. The same process is done for all other parts of the simulator such as the yoke on the same figure above.

4.3 Testing the Simulator

At this point it may be possible to test the modelled cockpit in Xplane and the first thing is to export the 3D model from Blender to a compatible format that can be read by Xplane which is .obj. Both the .blend and .obj files must be in the same directory as the .acf with the name:

airplanename_cockpit, to elaborate; the .acf file is ERJ145.acf, and the .blend and .obj files are ERJ145_cockpit.blend / .obj.

Figure 4-13: Testing the Simulator in Xplane

Figure (4-13) shows parts of the 3D model with live instruments, PFD, MFD, EICAS, the yoke, pedals, centre pedestal panel, and the overhead panel. The instruments and displays are mapped using the panel region and

UV mapping, the landing gear is animated by using the armature, datarefs and bones.

In figure (4-14) which is a magnified excerpt from the figure (4-13) showing the artificial horizon, the airspeed indicator, the altimeter, and the two engines percentage of power, all the indicators where working during the flight test.

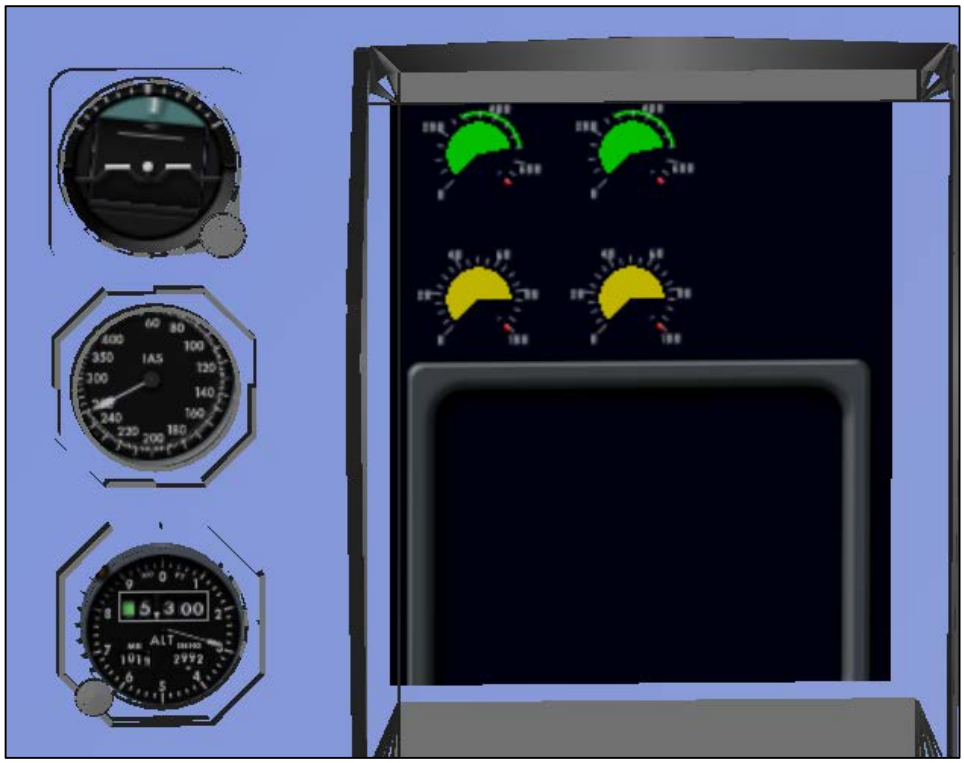


Figure 4-14: Artificial Horizon, Airspeed, Altimeter, and Engines Indicators

The test was conducted with cross-reference to some the International Civil Aviation Organization (ICAO) FSTD qualifications criteria in terms of engines, control dynamics, ground effect, sound system, visual system and motion system. For the engine: tests are required to show the response of the critical engine parameter to a rapid throttle movement for an engine acceleration and an engine deceleration [23] as shown in figure (4-15).

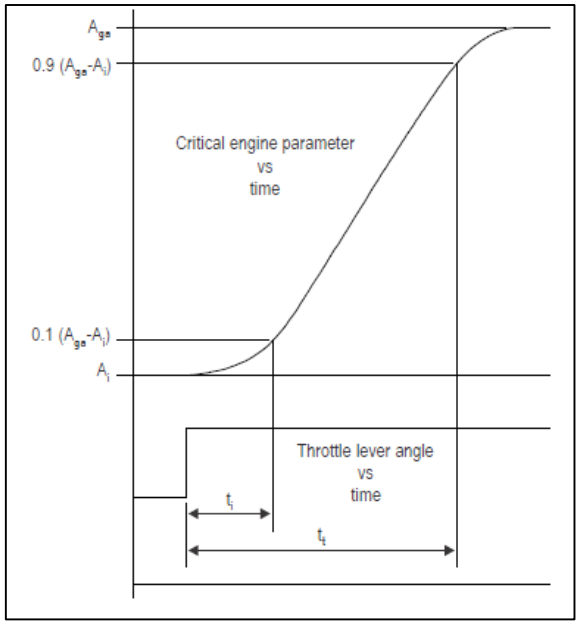
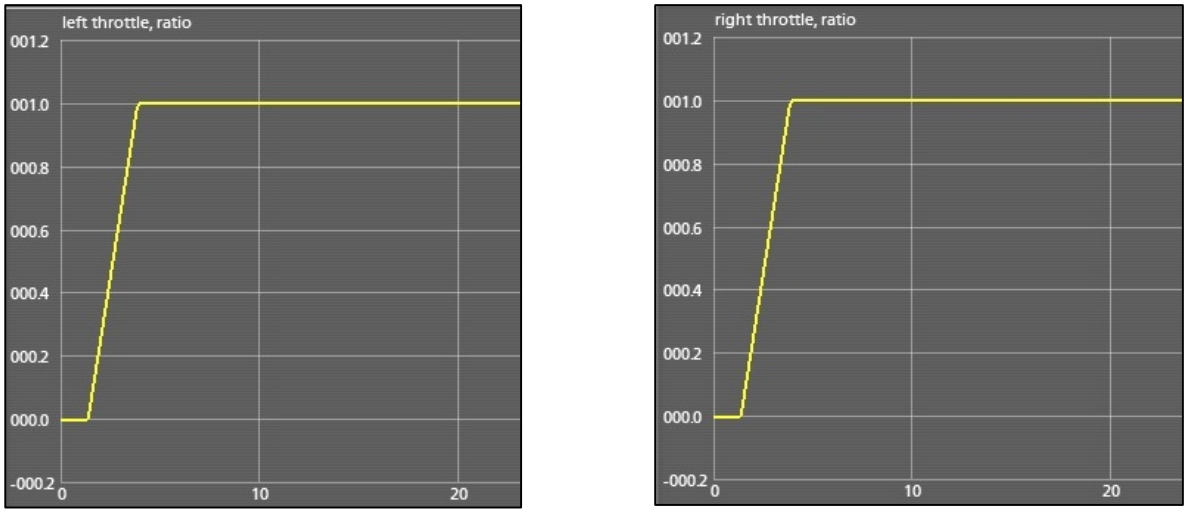


Figure 4-15: Engine Acceleration Criteria [23]

Where: A_{go}: critical engine power at go-around power, A_i: critical engine parameter at idle power, t_i: total time from initial throttle movement until a critical engine parameter reaches 10% of its total response above idle power t_t = total time from initial throttle movement until a critical engine parameter reaches t 90% of its total response above idle power.

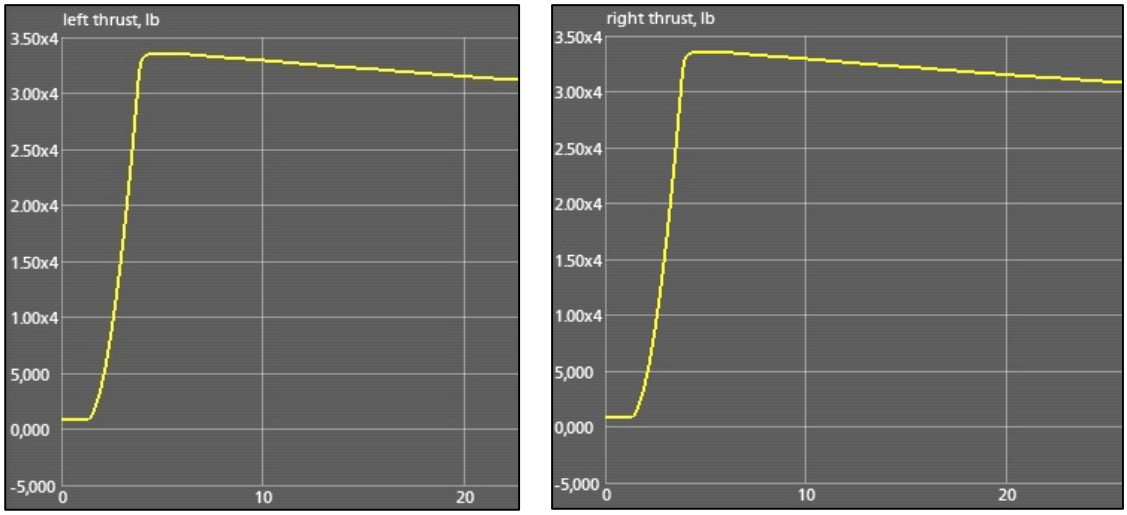


(a) (b)

Figure 4-16: Engine Acceleration Criteria for (a) Left and (b) Right Engines

The test was in compliance with criteria for acceleration and also with deceleration; which is in same parameters with acceleration but in reverse and as seen in figure (4-16) both engines follow the same pattern as the criteria depicted in figure (4-15) starting from t_i up to A_{go}.

Another test was conducted by the comparison of the maximum thrust of the engines produced by the simulated model and the engines data described in Appendix (A); the maximum thrust produced by the series of engines is given between 6000 to 8000 lbf depending upon the installed engines. The result given by the model is >6000 and <6500 lbf for both left and right engines as shown in figure (4-17).



(a) (b)

Figure 4-17: Maximum Thrust for (a) Left and (b) Right Engines

For the V_2 speed for the ERJ; it's indicated as 130 Knots Indicated Airspeed (KIAS) [24], in figure (4-18) it is depicted that the indicated air speed is about 128 Kias when the vertical velocity indicated (VVI) begins to have a positive rate of climb while the altitude is leveling from 0 ft.

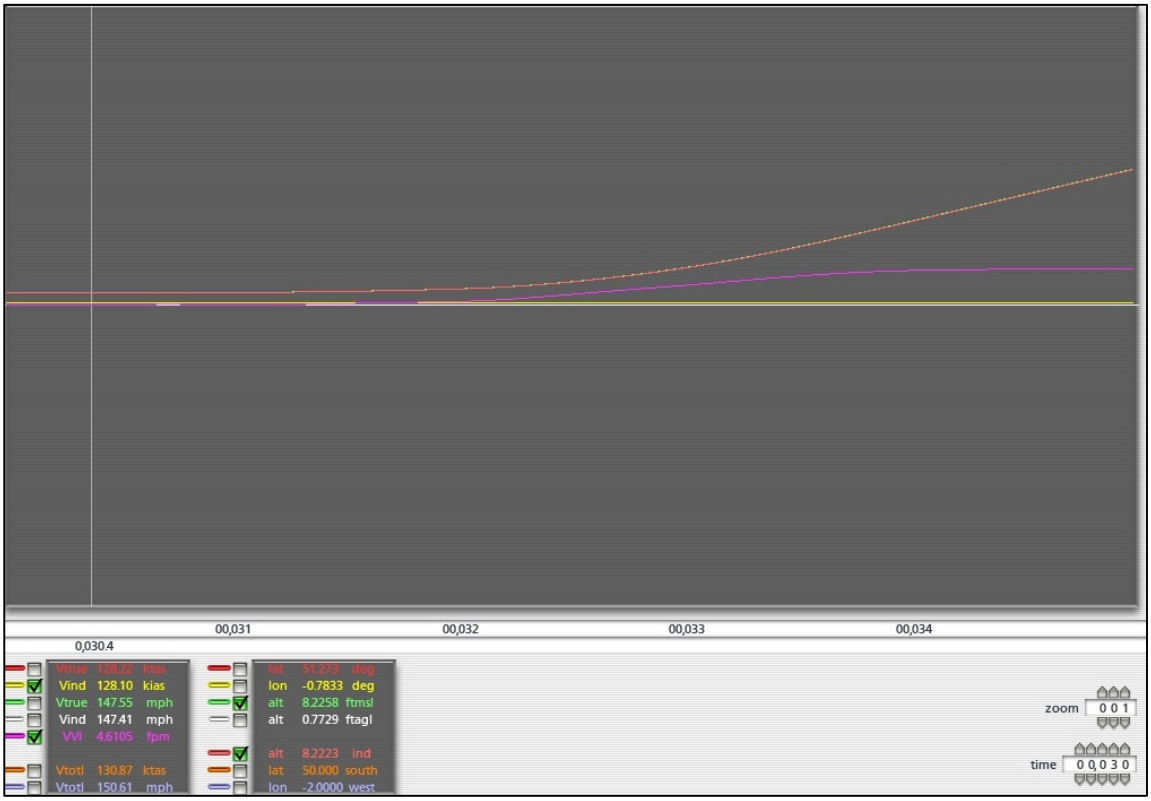


Figure 4-18: Knots Indicated Airspeed, Vertical Velocity and Altitude

As for the visual system we can state that the simulator met this criterion; height and Runway Visual Range (RVR) for the assessment have been selected in order to produce a visual scene that can be readily assessed for accuracy (RVR calibration) and where spatial accuracy (center line and Glide Slope (G/S)) of the airplane being simulated can be readily determined using approach/runway lighting and flight deck instrument.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion:

This Research outlines the importance of flight simulators (FS) in the process of pilot training, engineers, and technicians. A newly developed approach of designing FS is the integration with the VR technology meeting the evolution of the aircraft industry. So a closer look into the evolution of FS is discussed in excruciating details alongside the VR technology and its variants. Then an aircraft has been modelled which is an Embraer ERJ-145 (Embraer Regional Jet) in a 3D modeling software Plane Maker with Blender 2.45 and tested in a visual engine namely X-Plane v9.

The modelled cockpit was a replica of the real cockpit using photographs of the cockpit, and it's possible to model any other cockpit of whether fixed or rotatory wing air vehicle.

Then the cockpit was integrated in the visual engine, and it's notable that the integration needs third-party software or plugin (Xplane2Blender) for a complete and error-free integration. A test has been conducted to check how the modelled aircraft interacts with the visual engine, and the dynamic response of the aircraft was found to be acceptable.

The primary results show a promising an acceptable future for the VRFS, where currently the VRFS is used for engineering purposes only not for pilot training.

The training experience varies greatly between physical and virtual flight simulators; where some features can't be found on physical simulators and to be found on the virtual and vice versa.

5.2 Recommendations:

For further integration for immersive VR experience, the use of VR goggles is very much recommended along with other input devices such as gloves.

As for the graphics and terrains, the addition of real-time graphics where pilots can select the map they want to fly at, to some extent at relatively real-time. This also presents a valuable contribution.

This project can also be enhanced by the merging of multi-contributors acting as a combined network for more realistic experience or by combing it to the Virtual Air Traffic Simulation network (VATSIM) where people from around the world flying online or acting as virtual ATC.

REFERENCES

[1] Steven M. LaValle, Virtual Reality, Cambridge University Press, 2015.

[2] Maithili Shah, Parth Mehta, and Neha Katre, A Review of New Technologies: AR, VR, MR, International Journal of Computer Applications, August 2017.

[3] Zhigeng Pan, Adrian David Cheok, Hongwei Yang, Jiejie Zhu, and Jiaoying Shi, Virtual Reality and Mixed Reality for Virtual Learning Environments, Computers & Graphics, 2006.

[4] Rustin Webster, Alex Clark, Turn-Key Solutions: Virtual Reality, Proceedings of the ASME 2015 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, August 2015.

[5] Burdea, G. C., and Coiffet, Virtual Reality Technology, John Wiley & Sons, 2003.

[6] Sutherland, I. E., Sketchpad, A Man-Machine Graphical Communication System, Department of Electrical Engineering, M.I.T., 1963.

[7] Turgay Aslandere, Daniel Dreyer, Frieder Pankratz, and Renè Schubotz, A Generic Virtual Reality Flight Simulator, Technische Universität München (TUM), September 2018.

[8] Kelvin Valentino, Kevin Christian, and Endra Joelianto, Virtual Reality Flight Simulator, Internetworking Indonesia Journal Vol. 9/No. 1, 2017.

[9] Tomasz Mazuryk and Michael Gervautz, Virtual Reality, History, Applications, Technology and Future, Institute of Computer Graphics Vienna University of Technology, 1999.

[10] Panagiotis Drakopoulos, Virtual Reality Flight Simulator with Hand-Tracking Technology and Haptic Feedback, University of Greece, 2017.

[11] Norman I. Badler, and Andrew S. Glassner, 3D Object Modeling, Computer and Information Science Department, University of Pennsylvania, 1999.

[12] Derek O' Reilly, Computer Generation of Photorealistic Images using Ray Tracing, Dublin City University, 1991.

[13] https://blender.org/about/, [Accessed: 2nd of July, 2021].

[14] https://help.sketchup.com/en/sketchup/sketchup, [Accessed: 2nd of July, 2021].

[15] https://unrealengine.com/en-US/what-is-unreal-engine-4, [Accessed: 2nd of July, 2021].

[16] https://unity.com, [Accessed: 2nd of July, 2021].

[17] http://home.flightgear.org/about/, [Accessed: 2nd of July, 2021].

[18] https://x-plane.com/files/manuals/X-ne_10_Desktop_manual, [Accessed: 20th of February, 2019].

[19] https://www.prepar3d.com/product-overview/, [Accessed: 2nd of July, 2021].

[20] H. Scheirich: Stereoscopics - Principles and Techniques. Diploma Thesis, Vienna University of Technology, Austria (1994)

[21] Chris Totten, Game Character Creation with Blender and Unity, John Wiley & Sons, Inc., 2012.

[22] https://xp2b-docs.gitbook.io/xplane2blender-docs/index, [Accessed: 2nd of July, 2021].

[23] Manual of Criteria for the Qualification of Flight Simulation Training Devices, Doc 9625, Volume 1 – Aeroplanes, Fourth Edition, International Civil Aviation Organization, 2015.

[24] https://www.embraercommercialaviation.com/commercial-jets/erj145/, [Accessed: 22nd of November, 2021].

[25] https://www.vatsim.net, [Accessed: 2nd of July, 2021].

[26] https://developer.x-plane.com/manuals/planemaker, [Accessed: 2nd of July, 2021].

[27] Federal Aviation Administration (FAA), Advanced Avionics Handbook, U.S. Department of Transportation, 2009.

[28] https://360cities.net/image/cockpit-embraer-erj-145-airplane-usa, [Accessed: 2nd of July, 2021].

[29] https://developer.x-plane.com/2009/04/datarefs-vs-commands-i-whats-the-difference, [Accessed: 13th of October, 2019].

[30] https://xp2b-docs.gitbook.io/xplane2blender-docs/index-3/34_object_ settings, [Accessed: 13th of October, 2019].

[31] https://docs.blender.org/manual/en/latest/animation/armatures/introduction, [Accessed: 13th of October, 2019].

APPENDICES

APPENDIX (A)

THE ERJ DATA

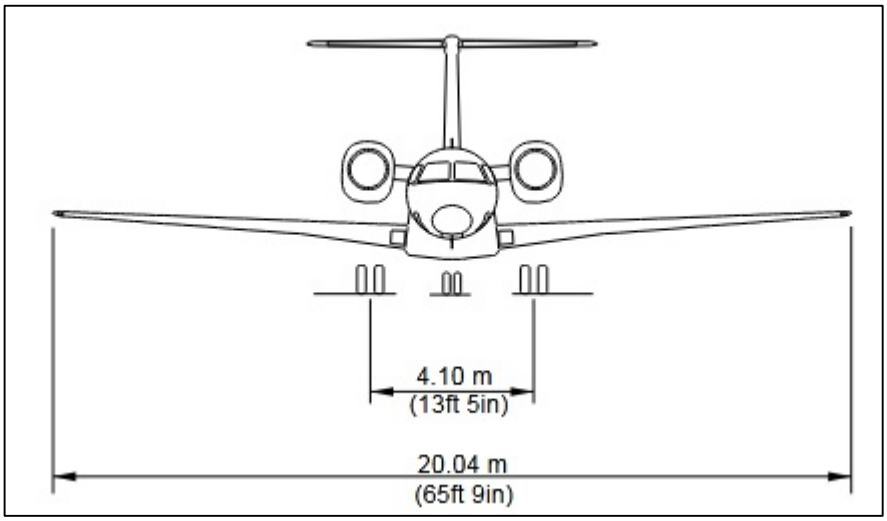
Airplane General Characteristics

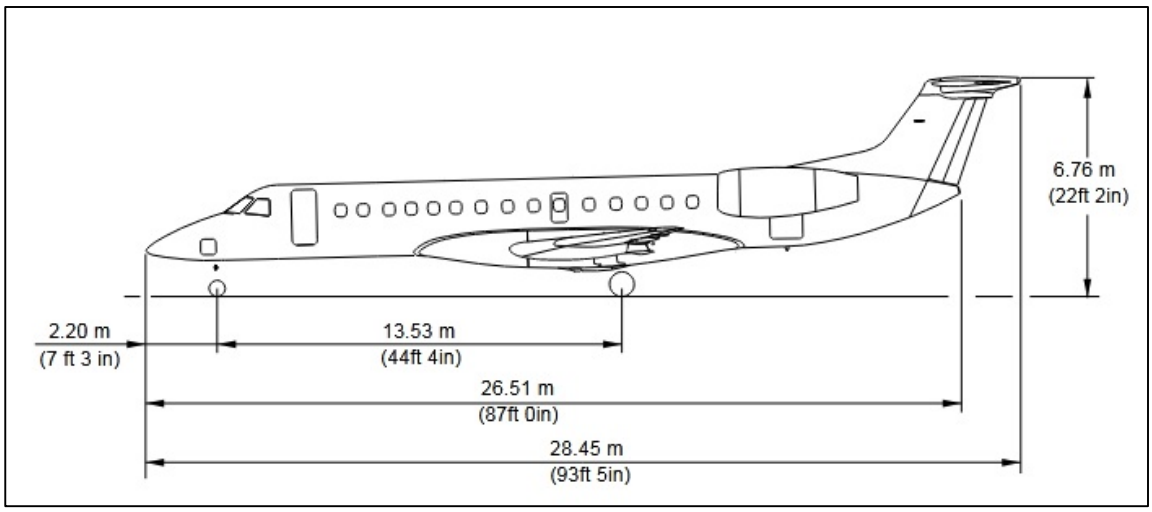
	EP		LR	
Maximum Takeoff Weight	20,990 kg	46,275 lb	22,000 kg	48,501 lb
Maximum Landing Weight	18,700 kg	41,226 lb	19,300 kg	42,549 lb
Basic Operating Weight (std)	11,947 kg	26,339 lb	12,114 kg	26,706 lb
Maximum Zero Fuel Weight	17,100 kg	37,699 lb	17,900 kg	39,462 lb
Maximum Payload	5,153 kg	11,360 lb	5,786 kg	12,755 lb
Maximum Usable Fuel	5,146 l	1,359 gal	6,396 l	1,690 gal

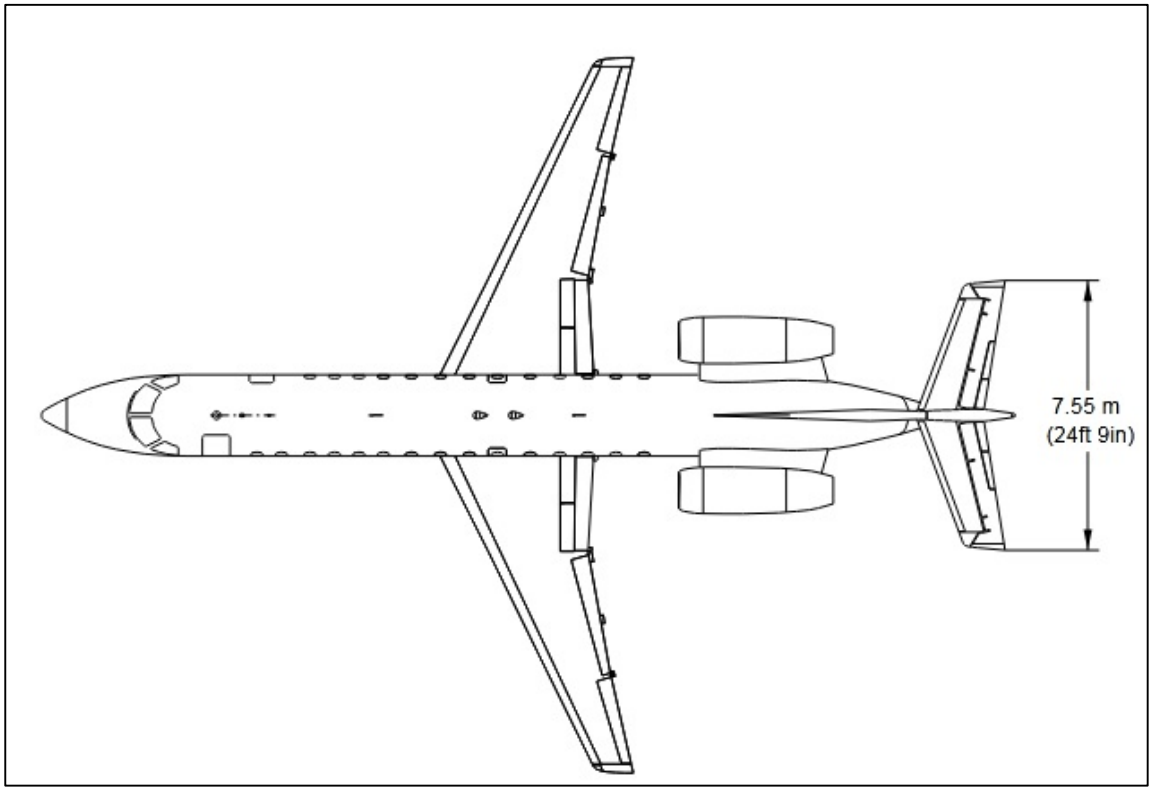
Airplane Dimensions

External Dimensions	
Overall span	20.04 m (65 ft 9 in)
Height (maximum)	6.76 m (22 ft 2 in)
Overall length	28.45 m (93 ft 5 in)
Wing	
Reference area	51.18 m^2 (551 ft^2)
Reference aspect ratio	

Fuselage	
Total Length	26,51 m (87 ft 0 in)
Length of pressurized section	19.67 m (64 ft 6 in)
Outside diameter	2.28 m (7 ft 6 in)
Horizontal Tail	
Span	7.55 m (24 ft 9 in)
Area	11.20 m^2 (120.6 ft^2)
Vertical Tail	
Reference area	7.20 m^2 (77.5 ft^2)

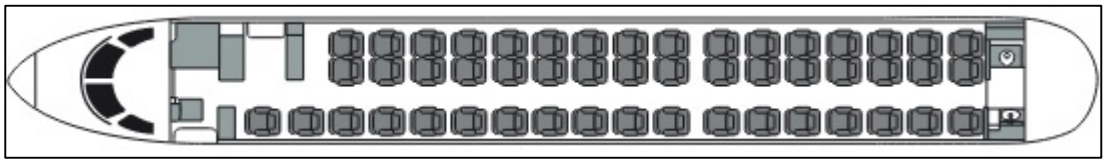






Interior Configuration

50 Seats at 31” Pitch



Interior Arrangements

The standard interior arrangement provides accommodation for two pilots, one flight observer, one flight attendant, and 44 passengers. One additional flight attendant seat is available as an option.

Cockpit

The ″quiet and dark″ cockpit is designed to accommodate the pilots with comfort during all flight phases, with minimum workload and maximum safety. The cockpit is provided with two pilot seats, a foldable flight observer seat, control columns and pedals, control pedestal, left, right, and aft consoles, as well as main, overhead, circuit breaker, and glareshield panels. A sunshade is provided for each pilot and the compartment is separated from the passenger cabin by a partition with a lockable door.

Panels

The main instrument panel displays the main navigation, engine, and system indications, through the PFD, MFD, and EICAS displays, the audio selection, ELT reset, and the landing gear and pedal electric adjustment controls. It also accommodates the standby instruments and displays reversionary functions.

A glareshield panel is located over the main panel, including the master caution and master warning lights, flight control, display control, and lighting

intensity controls. One of the different possible configurations of glareshield panel includes dual radar control panels.

An overhead panel provides the hydraulic, electrical, powerplant, APU, fire protection, environmental, and external and internal lighting controls. The circuit breakers, in ordered and grouped positions, are placed on a panel aft of the overhead panel.

Left and Right Consoles

The left and right consoles accommodate the nose wheel steering handle, ashtrays, holders for cups, headset, and microphone, oxygen masks and oxygen control, a waste container, rechargeable flashlight, and recesses for crew publications.

Control Pedestal

The control pedestal, located between the two pilots, presents the engine control levers, the engine thrust rating panel, the speed brake lever, the emergency/parking brake lever, flight control switches (including flap selection), the pressurization control, the EICAS reversionary panel, radio management units, single radar control panel, HF control (optional), aileron/elevator disconnect handles, AP control, SPS, T/O configuration switch, and an FMS control display unit.

Pilot Seat

The pilot seat is provided with longitudinal, vertical (electrically actuated), seat back, and lumbar adjustments. The seat is attached to tracks which permit the horizontal adjustments. An extended longitudinal travel permits pilot rest during long cruise flights (pilot foot rests are provided at the bottom of the main instrument panel).

Engines:

Two high bypass ratio rear-mounted engines.

Type:

AE 3007

Models:

AE 3007C, AE 3007A, AE 3007A1/1, AE 3007A1, AE 3007A1/2, AE 3007A1/3,AE 3007A3, AE 3007A1P, AE 3007C1, AE 3007A1E, AE 3007A2, AE 3007C2.

Manufacturer:

Rolls-Royce Corporation P.O. Box 420 Indianapolis, Indiana 46206-0420, United States of America.

Description:

Direct drive turbofan engine of modular design. Incorporates a single stage fan, a 14-stage axial compressor with 6 stages of variable vanes (including inlet guide vanes), an annular combustion chamber, a two-stage high pressure turbine and a 3-stage low pressure turbine. The accessory gearbox is mounted at the bottom of the engine. The engine is equipped with two single channel Full Authority Digital Engine Control (FADEC) System units which are mounted in the aircraft. The engine features fore and aft mounting provisions, which allow either underwing pylon or aft fuselage mounting installation.

Dimensions:

Overall Length	2.92 m (115.08 inches)

Overall Height	1.41 m (55.70 inches)
Overall Width	1.17 m (46.14 inches)

Ratings:

Model	Static Thrust		Flat Rated
	Take-Off (5min.)	**Maximum Continuous**	
AE 3007C	28.65 kN (6442 lbf)	28.65 kN (6442 lbf)	ISA+15°C
AE 3007A AE 3007A1/1	33.71 kN (7580 lbf)	30.33 kN (6820 lbf)	ISA+15°C
AE 3007A1/2	33.71 kN (7580 lbf)	30.33 kN (6820 lbf)	ISA+23°C
AE 3007A1	33.71 kN (7580 lbf)	30.33 kN (6820 lbf)	ISA+30°C
AE 3007A1/3	33.71 kN (7580 lbf)	30.33 kN (6820 lbf)	ISA+30°C
AE 3007A3	32.02 kN (7201 lbf)	30.33 kN (6820 lbf)	ISA+15°C
AE 3007A1P	37.08 kN (8338 lbf)	30.33 kN (6820 lbf)	ISA+19°C
AE 3007C1	30.08 kN (6764 lbf)	30.08 kN (6764 lbf)	ISA+15°C
AE 3007A1E	39.67 kN (8917 lbf)	32.65 kN (7339 lbf)	ISA+19°C (Take-Off) ISA+30°C (Max Continuous)
AE 3007A2	41.99 kN (9440 lbf)	36.02 kN (8097 lbf)	ISA+15°C (Take-Off) ISA+20°C (Max Continuous)

AE 3007C2	31.32 kN (7042 lbf)	31.32 kN (7042 lbf)	ISA+15°C

APPENDIX (B)

Xplane 2 Blender Documentation

Preface

This is the final version of Xplane2Blender v 3.10. It's been tested and works with Blender 2.49 on Mac, Windows, and Linux. Development of this script is no longer done by Marginal, but collectively and in an Open-Source manner. So please file requests and issues at the project website.

Overview

Blender is an open source 3D object editor for Windows, Mac and UNIX.

These Blender scripts export models created in Blender to X-Plane v7, v8, v9 or CSL .obj format.

The scripts also import existing X-Plane v6, v7, v8, v9 and CSL .obj files and X-Plane v7, v8 and v9 .acf airplanes into Blender.

Requirements

Runs on Windows 2000 or later, Mac OS 10.3.9 or later, and Linux.

Importing objects

First, move the 3D Cursor to where you want the imported object to be placed. Usually you'll want the object to be placed at the origin so just press Shift C to centre the 3D Cursor at the origin.

Choose File → Import → X-Plane Object, select a .obj file and press Import OBJ.

The scenery is imported at the 3D Cursor position.

You can import multiple .obj files and re-export them as a single file. But note that the X-Plane .obj file format only supports the use of one texture file, so you'll have to create a single larger file containing all required textures - see below. Or you can have Blender create this single file automatically by selecting all the objects and, in a UV/Image Editor window, choosing Image → Consolidate into one image.

Importing planes

First, move the 3D Cursor to where you want the imported plane or weapon to be placed. Usually you'll want it to be placed at the origin so just press Shift C to centre the 3D Cursor at the origin.

Choose File → Import → X-Plane Plane or Weapon and choose whether to import the plane or weapon so that the "reference point" is located at the 3D Cursor (for making cockpit & misc objects) or so that the "centre of gravity" is located at the 3D Cursor (for making CSLs and static scenery).

Then select a .acf or .wpn file and press Import ACF or WPN.

The script creates up to three versions of the plane or weapon in Blender, one in each of layers 1, 2 and 3. The versions in layers 2 and 3 use approximately 1/10th and 1/100th of the number of faces compared to the version in layer 1. See "Level of Detail" below for an explanation of why it does this.

Imported planes need some tweaking before you can export them as scenery or as a CSL object; see "Tweaking planes" below.

Exporting objects

First, choose File → Save As… and save the. blend file in the aircraft or scenery folder where you want the .obj file to end up. Then choose:

File → Export → X-Plane CSL Object or
File → Export → X-Plane v7 Object or

`File → Export → X-Plane v8/v9 Object`

The object is exported in the same folder and with the same name as the current Blender file, but with a .obj extension. Blender may display some informational messages - click on one of these messages to see which object(s) the message refers to.

If there is an error then the scripts will attempt to identify and highlight the offending Blender object(s).

cs: These objects are intended for use with multi-user plugins such as XSquawkBox or X-IvAp. v8: These objects are supported by X-Plane versions 8.20 and later. v9: These objects are supported by X-Plane versions 9.00 and later.

Creating X-Plane scenery

- Find and open the Custom Scenery folder inside of your X-Plane installation.
- Create a subfolder with the name of the scenery package that you're making.
- Save your Blender file in this subfolder folder with a descriptive name, eg: X-Plane/Custom Scenery/EGLL/house.blend
- `The X-Plane .obj file will be exported to the same place, ie:`

  `X-Plane/Custom Scenery/EGLL/house.obj`

Tweaking planes

Imported planes need to be positioned correctly on the ground for use as static scenery (don't do this if you're making a CSL or cockpit object):

- Select layers 1-3, select all objects and position the plane so that its undercarriage is sitting directly on the ground (represented by the x/y axes).
- You may also need to rotate the plane slightly so that all wheels are level; press r and move the mouse to rotate the plane so that its undercarriage is sitting directly on the ground. Click to set the plane's position.

The primary file for textures is named airplane_paint. Most planes also use textures from a secondary file named airplane_paint2. Objects that use textures from the secondary file are imported with "*" after their name to make them easier to identify in Blender's Outliner window.

The X-Plane .obj scenery file formats only support the use of a single file for textures, up to 1024x1024 pixels in size. If your plane only has a few simple objects that use textures from airplane_paint2 then you should re-texture these objects to use airplane_paint, following the same procedure described below for weapons and misc objects. If that is not feasible you can use this procedure to make the plane use textures from a single file:

- Save your Blender model.
- In an image editor application, resize airplane_paint and airplane_paint2 to 512x512. (You don't need to save these resized versions).
- Create a new bitmap file 1024 pixels wide and 512 pixels high.
- Paste the resized airplane_paint into the left half of this bitmap.
- Paste the resized airplane_paint2 into the right half of this bitmap.

- Save the bitmap file in the same folder and with the same name as your .blend file. If you're making a CSL object then you must save in PNG format.
- If your plane uses night-time textures then repeat this procedure for the _LIT bitmap files.
- In a Blender UV/Image Editor window choose Image → Merge _paint and _paint2.

If your imported plane uses weapons or misc objects then each of these will use an additional bitmap file. Weapons are imported with their names starting with "Wnn" and objects with their names starting with "Onn". Also note that reduced-LOD versions of weapons and misc objects may be present in layers 2 and 3.

Open an Outliner window and choose View → Show Outliner. For each mesh that has a name starting with "Wnn" or "Onn" or ending with "*", either:

- Delete the mesh, or
- Copy the required textures to an unused area in the primary bitmap file and use the UV/Image Editor window to map the new copy of the textures to the mesh's faces.

Consider performance issues when the plane is rendered in X-Plane. Ask yourself the following questions:

- Most important: Do you really need fully detailed 1024x1024 textures for your plane? Video memory is used up by terrain and object geometry and textures. Once you run out of video memory the GPU has to fall back to main memory, and this really slows things down.

One 1024x1024 texture uses 4MB of video memory. Some people are running X-Plane on computers with only 32MB of video memory, so one 1024x1024 texture at "extreme res" in X-Plane uses 1/8th of their video memory. Consider resizing the texture file in an image editor program to half or even quarter size. Use Image → Replace to use the new texture file.

- Does the model have hidden faces? Some versions of X-Plane don't handle hidden faces in v7 scenery objects very well and they also a cause a small performance hit. Look for things like wings or misc bodies that are partially or wholly buried in the fuselage and delete any wholly hidden faces before exporting.
- Do you really need all that detail? Consider deleting details like flap tracks, antennae etc before exporting. This especially goes for the lower Level of Detail versions of the plane in layers 2 and 3 which are only viewed in X-Plane from >1000m and >4000m respectively.

Creating 3D cockpits

X-Plane 3D cockpits are just normal v7, v8 or v9 scenery objects except that cockpits can't contain multiple Levels of Detail, so only objects in Blender layer 1 are exported. If you want to keep objects in your Blender scene for reference but which you don't want to export - eg the plane fuselage - then put them in layer 4 or greater before exporting.

Choose File → Save As and save the blender file in the same folder as your plane's .acf file with the name airplane_cockpit.blend, airplane_cockpit_INN.blend or airplane_cockpit_OUT.blend (where airplane is the name of your plane's .acf file):

- X-Plane displays airplane_cockpit.obj in both internal and external views.
- X-Plane displays airplane_cockpit_INN.obj only in internal views.
- X-Plane displays airplane_cockpit_OUT.obj only in external views. If you create an airplane_cockpit_INN.obj and/or airplane_cockpit_OUT.obj then you should not create airplane_cockpit.obj.

You will usually want to import your plane into Blender to act as a reference and/or starting point for your cockpit. Delete any plane parts that you don't need in creating your cockpit - you only need to keep the fuselage itself plus any relevant Misc Bodies. After import, your cockpit uses the same texture file as your plane, ie airplane_paint. Choose Image → Replace in a UV/Image Editor window to use a different texture file, which can be named anything you like (but no spaces) and which should live in the same folder as your plane's .acf file.

To make your 3D cockpit appear in X-Plane, on the Standard → Viewpoint → View screen in PlaneMaker, check the show cockpit object in: INSIDE views, exact forwards option. To hide the 2D cockpit altogether, also check the show cockpit object in: PANEL views, exact forwards option; hiding the 2D cockpit means that you no longer have to leave a large part of the Panel Texture transparent to represent the windscreen, which gives you more room on the Panel Texture for instruments.

Cockpit instruments

To construct moving cockpit instruments paint the …/cockpit/-PANELS-/Panel.png texture in an image editor application and place

instuments in PlaneMaker as as you would for a 2D panel (but bear in mind that only the top 768 lines of the Panel Texture can be used in the 3D cockpit in X-Plane versions prior to 8.20). The Panel Texture can by 1024×any size in X-Plane v8, and any size up to 2048×2048 in X-Plane v9. Normally you can only use a single file to texture your X-Plane objects. But when constructing a 3D cockpit you can additionally use this …/cockpit/-PANELS-/Panel.png file - the instruments that X-Plane draws on the 2D panel will also appear in your 3D model.

The …/cockpit/-PANELS-/Panel.png texture doesn't contain any instruments when you load it into Blender (unless you've painted them on yourself). This makes it hard to tell in Blender where X-Plane will draw the instruments. So it's easier if you use a screenshot of the panel with the instruments drawn on it, instead of the real Panel Texture. If your display is larger than your Panel Texture, then this is simple:

- Run PlaneMaker.
- Choose Background → Rendering Options and set the size to be equal to the size of your Panel Texture.
- Restart PlaneMaker
- Choose Standard → Panel
- Take a screenshot: Press Alt PrintScreen (PC) or Command Shift 3 (Mac).
- Paste (PC) or load (Mac) the screenshot into an image editor application.
- Crop the window borders etc from the screenshot so that the image is exactly the same size as your Panel Texture.

- Save the screenshot as ScreenshotPanel.png (or any filename ending in panel.) in the same folder as your plane's .acf file.
- Use the ScreenshotPanel.png texture on those faces that you want to display moving cockpit instruments in X-Plane.
- The screenshot file does not need to be distributed with your finished plane.

If your Panel Texture is larger than your display then you cannot take a screenshot of the whole panel. In this case you'll need to take multiple screenshots of the panel in PlaneMaker and stich them together in an image editing application.

If you later want to resize your Panel Texture then use the procedure described below.

Note that X-Plane versions prior to 8.20 only display the 3D cockpit when running at the default 1024x768 resolution. You may want to mention this in the Readme with your plane if your plane is intended to work in X-Plane versions prior to 8.20.

v9: Cockpit Panel Regions

The cockpit Panel Texture uses a lot of video memory, much of which is wasted when the 3D cockpit is being displayed:

- X-Plane has to round up the height and width of your Panel Texture to be powers of two. eg if your Panel Texture is 1600×1200 pixels then X-Plane rounds this up to 2048×2048 pixels, which requires 16MB of video memory. More if you also supply a LIT Panel Texture.

- Typically up to half of your Panel Texture represents your plane's windscreen, which is fully transparent. You can't make use of this part of the texture in any useful way in a 3D cockpit, so the memory that it consumes is wasted. (Note: You can construct a tinted windscreen in your 3D cockpit quite cheaply by using a small semi-transparent part of the non-panel texture).
- The Panel Texture contains an alpha channel for transparency. The alpha channel accounts for ¼ of the memory that the texture consumes. But often your only need for transparency in the Panel Texture is to represent the 2D windscreen, which is of no use in a 3D cockpit, so the memory that the alpha channel consumes is wasted.

A "Panel Region" is a new texture which is cut out from your Panel Texture:

- You can create up to 4 Panel Regions (which can overlap).
- The height and width of a Panel Region texture must be a power of two eg 128, 256, 512, 1024 or 2048, but it doesn't have to be square.
- Panel Region textures are opaque - they don't contain an alpha channel.

When you use Panel Regions instead of the Panel Texture to texture your 3D cockpit, X-Plane discards the Panel Texture's alpha channel and also discards all areas of the Panel Texture other than the pieces that you cut out to make the Panel Regions. This reduces video memory requirements and improves performance.

Creating a Panel Region

- In the UV/Image Editor window, select your Panel Texture from the pop-up menu.
- Choose Image → X-Plane panel regions → Create new region
- Enter the co-ordinates in your Panel Texture where you want the bottom-left pixel of the new Panel Region to start, and the width and height of the new Panel Region.

Any faces that you've textured using the Panel Texture which are contained inside the new Panel Region are transferred over to use the new Panel Region.

Any areas that are fully transparent in the Panel Texture are coloured sky blue in the new Panel Region. You'll get undefined (ie weird) results in X-Plane if you use these sky blue areas to texture your faces.

(Note: When you create a Panel Region, Blender also creates a hidden object named "PanellRegionHandler" to store accounting information. Don't mess with this object).

Deleting a Panel Region

- In the UV/Image Editor window, select your Panel Region from the pop-up menu.
- Choose Image → X-Plane panel regions → Delete this region

Any faces that you've textured using this Panel Region are transferred back to using the Panel Texture.

The deleted Panel Region will remain in the UV/Image Editor window's pop-up menu for a while until Blender figures out that it can remove it. But the deleted Panel Region won't count towards your maximum of four Panel Regions.

Re-loading the Panel Regions

The Panel Regions aren't automatically updated when you edit your Panel Texture in an image editor application and then reload it in Blender, or when you reload your .blend file.

- In the UV/Image Editor window, select your Panel Texture from the pop-up menu.
- Choose Image → X-Plane panel regions → Reload all regions

Using Blender to create X-Plane objects

Only Lamps and Meshes are exported to X-Plane. You can use other Blender object types, eg Curves and Surfaces, to construct your scenery as long as you convert them to meshes before exporting to X-Plane.

Lamps

Only Lamp objects of type "Lamp" are exported to X-Plane. Lamp objects of types "Area", "Spot", "Semi" and "Hemi" are ignored (and so can be used to illuminate your model in Blender).

v8/v9: Lamp objects with certain words in their names have special behaviours when exported to an X-Plane v8 or v9 object:

- Lamp - Normal (legacy) light. The colour is determined by the R,G,B sliders on the Lamp panel (F5).
- Flash - Flashing (legacy) light. The colour is determined by the R,G,B sliders on the Lamp panel (F5).
- Traffic, smoke_black, smoke_white - As for X-Plane v7 objects; see below. (R,G,B settings are ignored).
- other - X-Plane pre-defined "named" or "custom" light. (Supported by X-Plane 8.50 and later. R,G,B settings are ignored). The name of the X-Plane light is taken from the value of a property named name if this exists, otherwise from the name of the lamp object.

v7: Lamp objects with certain words in their names have special behaviours when exported to an X-Plane v7 object:

- Flash - Flashing light. The colour is determined by the R,G,B sliders on the Lamp panel (F5).
- airplane_beacon - Red pulsing anti-collision light. (R,G,B settings are ignored).
- airplane_strobe - White strobe light. (R,G,B settings are ignored).
- Traffic - Cycles red, orange, green. (R,G,B settings are ignored).
- smoke_black or smoke_white - Not really a light; emits smoke. The size of the smoke puffs is determined by the Energy slider on the Lamp panel (F5) (R,G,B settings are ignored).
- other - Normal light. The colour is determined by the R,G,B sliders on the Lamp panel (F5).

csl: Lamp objects with certain names have special behaviours when exported to an X-Plane CSL object. The XSquawkBox documentation strongly recommends that you use these special lights:

- airplane_landing - White landing light. (R,G,B settings are ignored).
- airplane_taxi - White taxi light. (R,G,B settings are ignored).
- airplane_nav_left - Red navigation/position light. (R,G,B settings are ignored).
- airplane_nav_right - Green navigation/position light. (R,G,B settings are ignored).
- airplane_beacon - Red pulsing anti-collision light, on when engines are running. (R,G,B settings are ignored).
- airplane_strobe - White strobe light. (R,G,B settings are ignored).
- other - Normal light. The colour is determined by the R,G,B sliders on the Lamp panel (F5).

v8/v9: Custom lights (supported by X-Plane 8.50 and later) are created using the vertices from a Mesh object: In the Material buttons panels (F5) add a new material to the mesh, then press the Halo button on the Links and Pipeline panel. You should use just one material.

- The Halo button and the R,G,B,A sliders on the Material panel control the light's R,G,B and A values. Alternatively you can create property named R, G, B and/or A to set these values.
- The HaloSize control on the Shaders panel controls the light's S value.

  Also press the HaloTex button on this panel to make Blender render the light correctly.

In the Texture buttons panels (F6) add a new texture to the material, and change the Texture Type to Image. You should use just one texture.

- On the Image panel load the texture file that contains the light that you want to use.
- On the Map Image panel press the UseAlpha button.

  Use the MinX, MinX, MaxX, MaxY settings to select a subset of the texture.

To drive the custom light using a dataref add a String property named name.

Meshes

Create faces with 3 or 4 edges (called "tri"s and "quad"s in X-Plane).

In the Link and Materials Editing panel (F9):

- Set Smooth and Set Solid control whether to smooth edges of faces in a mesh. This is useful when using multiple faces to simulate a curved surface. Go to "Object Mode", select the mesh and press Set Smooth. The effect is only visible in Blender 3D View windows when the Viewport Shading button is set to Solid or Shaded.

  v7: Only faces that are part of a Strip will be smoothed when displayed in X-Plane.

In the Texture Face Editing panel (F9) available in UV Face Select mode:

- Tex button controls whether the face has a texture:

In an image editor application create one texture file that is like a "collage" of all of the textures that you need in your model. The height and width of the texture file must be a power of two eg 128, 256, 512, 1024 or 2048 (X-Plane v9 only), but it doesn't have to be square. (See …/Custom Scenery/KSBD Demo Area/KSBD_example.png for an example).

Save the texture file in 32bit or 24bit PNG format (ie with or without an "Alpha" channel) in the aircraft or scenery folder where you want the X-Plane .obj file to end up.

Use the UV/Image Editor window to control mapping of the textures to the face.

- Tiles button controls whether the face is rendered with "polygon offsetting" (ATTR_poly_os) in X-Plane. Press this button for faces that lie flat on the ground to prevent Z-buffer thrashing in X-Plane. Don't press this button for other faces. For best results with X-Plane versions prior to 8.50 you should ensure that objects that use polygon offsetting are listed first in your .env or .dsf scenery file.

  v7: "polygon offsetting" does not produce reliable results in X-Plane versions prior to 8.20.

  csl: This button has no effect when exporting CSL objects.

- Collision button indicates that the face is not "hard" (ie not "landable on") in X-Plane. Making faces hard is very expensive, so this button

should normally be pressed. Unpress this button only for things like helicopter landing pads.

(Note: The meaning of this button was changed in v1.50. Use this script to turn the Collision button back on for all faces).

v9: You can control whether it is possible to fly under this hard face; add a Bool property named deck and give it the value True. You can't fly under hard faces in X-Plane v8.

v8/v9: You can specify the surface type; add a String property named surface and give it the value water, concrete, asphalt, grass, dirt, gravel, lakebed, snow, shoulder or blastpad. The surface type is ignored by X-Plane versions prior to 8.50.

v7: Only faces with 4 edges (ie "quads") are exported as "hard".

v7: X-Plane versions prior to 8.20 have a bug where .obj files that contain hard faces must be placed in WorldMaker with an "object heading" of 0. Otherwise the "hard" part of the surface ends up in the wrong place.

csl: This button has no effect when exporting CSL objects.

*Twoside button controls whether one or both sides of the face are displayed. Unless you have a lot of double-sided faces it is cheaper to avoid this button and to use two single-sided faces back-to-back instead.

- Alpha button is a hint to Blender and to X-Plane that the face is transparent or translucent. Use this for outwards-facing transparent or translucent faces to instruct X-Plane to draw these faces last so that you can see other faces through them. Don't press this button for opaque (normal) faces. See drawing order for more fine-grained control over drawing order.

v8/v9: In the Material buttons panels (F5) you can change the way that the mesh reacts to light by specifying a material. Changing between materials in X-Plane is expensive so you should ensure that you only use a few materials in your model. Press Add New to create a new material, or choose an existing material (if any) from the drop-down list. You should use just one material. Only a few of Blender's many material buttons affect X-Plane:

- Col button and the R,G,B sliders on the Material panel control the diffuse colour of the faces. X-Plane combines the colours that are specified by the texture (if any) with this setting. The default X-Plane setting is 1, 1, 1 (white). However the default setting of a new material in Blender is 0.8, 0.8, 0.8.

  You should set this to 1, 1, 1 - control the diffuse colour of the faces by editing the texture file instead.

- Spec slider on the Shaders panel controls the specularity (shininess) of the faces. The default X-Plane setting is 0 (matt). However the default setting of a new material in Blender is 0.5.
- Emit slider on the Shaders panel controls the emissive brightness of the faces. Emissive faces give off light so the effects of this setting are

most obvious at night. It is usually cheaper and easy to use LIT_ textures to control night-time brightness instead of using this setting. However using this setting allows you to create faces that also give off light during the daytime, eg on overcast days, which LIT_ textures do not. The default X-Plane and Blender setting is 0 (not emissive).

- Mir button and the R,G,B sliders on the Material panel control the emissive colour of the faces.

Use this script if you need to reset all faces in the scene to standard settings (ie no polygon offsetting, not hard, single sided, not transparent).

You can add "modifiers" in the Modifiers panel to change the way that the Mesh appers. Some useful modifiers when modelling for X-Plane are:

- EdgeSplit - automatically sharpen edges between mesh faces (this only has an effect if you've used the Set Smooth button)
- Subsurf - produce a more detailed version of the mesh by subdividing faces
- Curve - bend the mesh along a curved path

Lines

Blender doesn't support Lines directly. Use a mesh with one 4-edged face instead. The pair of vertices at each end of the "line" must be within 0.1 units of each other. The face must be the only face in its mesh and must not have a texture assigned to it.

Assign a material to the face and use the Col button and the R,G,B sliders on the Material panel to control the colour of the line. Faces not linked to a material will be exported coloured grey.

v8/v9: Animation

You can make lamps, meshes and lines animate in X-Plane according to the value of any of the simulator datarefs listed here that have type "int", "float" or "double".

Basic animation

Create an "Armature" object. Make the lamps and/or meshes that you want to animate the children of the armature's "bone":

1. Click on the lamps and/or meshes (the "children")
2. Shift-click on the armature
3. Choose Pose Mode from the Mode menu in the 3D View window's toolbar
4. Click on the bone (the "parent")
5. Press Ctrl-P and select Bone from the popup menu

Once you have assigned a parent bone to your lamps/meshes, you can specify the simulator dataref that will drive the animation:

1. Choose Object Mode from the Mode menu in the 3D View window's toolbar
2. Select the child lamp or mesh
3. From the 3D View window's menubar choose Object → Scripts → X-Plane Animation
4. In the Parent Bone panel, use the pop-up menu to select the dataref

(or you can type in just the "leaf" name of the dataref into the text field, or the full name of a custom dataref)

5. Some datarefs require you to specify a "Part number";

   eg the dataref sim/flightmodel/engine/ENGN_thro represents the engine throttle settings, so you need to specify which engine you're referring to. Specify a "Part number" of 0 for the first engine, 1 for the second engine etc

6. Press the Apply button

Use frames to represent the desired position of the lamps/meshes at various dataref values - X-Plane will interpolate linearly between the positions:

1. In the Animation Frame field, in the Buttons window's menubar, select frame 1
2. Click on the armature
3. Choose Pose Mode from the Mode menu in the 3D View window's toolbar
4. Click on a bone
5. Move and/or rotate the bone
6. Press i and specify a LocRot key (ie location and rotation)
7. Repeat for any other bones in the armature
8. Select animation frame 2 and repeat

(Note: X-Plane's animation syntax is quite simple; so don't use IPOcurves, Vertex Groups, Deformations, Shape Keys or any other advanced Blender

animation techniques since these will be ignored by the exporter - only the positions specified by keys in the first n frames are significant to X-Plane.)

v8: X-Plane v8 only supports two frames, so if you want your .obj file to work in X-Plane v8 then you should insert "LocRot" keys only in frames 1 and 2. If you insert keys in frame 3 and above then your animation will not work at all in X-Plane v8. Use the Delete button in the X-Plane Animation dialog to delete any keys from additional frames.

v9: You can add "LocRot" keys in as many frames as you like. If you skip a frame then Blender and X-Plane will use the pose from the previous frame.

The X-Plane Animation dialog renames the parent bone to the "leaf" name of the dataref. So pressing the Draw Names button in the Armature panel can be helpful to see what's going on when you have lots of animations.

Use the Action Editor window to get an overview of which bones in the selected armature have keys inserted into which frames.

Controlling animation response to dataref values

By default, X-Plane will display the meshes in the frame 1 position when the dataref has a value of 0, and in the last frame position when the dataref has a value of 1. You can change these values:

1. Choose Object Mode from the Mode menu in the 3D View window's toolbar
2. Select the child lamp or mesh
3. From the 3D View window's menubar choose Object → Scripts → X-Plane Animation

4. Specify in the Frame #n fields the dataref values that correspond to each frame;

   eg the yoke pitch dataref yolk_pitch_ratio takes values between -1 (forward) and +1 (back) in X-Plane. So, to make X-Plane display the position in frame 1 when the yoke is pushed fully forward specify -1 in the Frame #1 field

5. Press the Apply button

v9: X-Plane will extrapolate your animation when the dataref has a value outside of the range that you specified in the Frame #n fields. You can stop the extrapolation and "clamp" your animation's position by repeating the poses and Frame #n values in the first two and/or the last two frames. Or you can cause your animation to loop back to frame 1 when the dataref value exceeds a certain number by specifying this number in the Loop field.

Using multiple datarefs

You can animate your lamps/meshes using multiple bones, each bone representing a different dataref:

1. In Edit Mode add additional bones to the armature.
2. Still in Edit Mode, in the Armature Bones panel, create parent/child relationships between each bone. (Note: This panel also lets you rename bones. Don't do this - use the X-Plane Animation dialog to name the bones after the datarefs that they represent).
3. Use the technique described above to make your lamps/meshes the children of the "youngest" bone in the chain.

4. Insert LocRot keys for every bone in the chain (each bone can have a different number of keys).

The X-Plane Animation dialog displays the settings for the lamp/mesh's parent bone, gandparent bone etc. (Note: Don't change the parent/child relationships between your lamps/meshes and their parent bones while the X-Plane Animation dialog is being displayed).

Hiding lamps and meshes

You can make all of the lamps and meshes in an animation disappear when a dataref is within a certain range:

1. Use the technique described above to make your lamps/meshes the children of an armature bone

   (If you don't want to animate your lamps/meshes then don't insert any animation keys for this bone, and the bone doesn't have to be a valid dataref)

   Choose Object Mode from the Mode menu in the 3D View window's toolbar

   Select a lamp or mesh that you want to hide

   From the 3D View window's menubar choose Object → Scripts → X-Plane Animation

In the last panel, press the Add New button

Specify the dataref and the range of values (it's OK to use datarefs that are not otherwise used in the animation)

You can make hidden lamps/meshes re-appear when a dataref value is within a certain range by adding another entry, and changing the type from Hide to Show. The dataref that you use to "show" the animation can be the same or different than the datarefs that you used to "hide" the animation. (The animation is always shown by default, so you only need to use a Show entry if you have used one or more Hide entries and you want to override them).

The order of Hide and Show entries is significant; the animation will be hidden if any of the Hide dataref values are in range, unless a subsequent Show dataref value is also in range. You can use the Up and Down buttons to change the order of the entries.

Note that the Hide and Show entries apply to all children of all bones in the armature. You can make an armature the child of a bone in a different armature; in which case all children are affected by any Hide and Show entries in parent armatures.

v8/v9: Drawing order

The order in which X-Plane draws the animations, lights, lines and triangles in your scenery or cockpit object usually has no effect on the appearance. So the exporter optimises the order of animations, lights, lines and triangles in your object to minimise the number of OpenGL state changes and therefore maximise X-Plane's framerate.

However drawing order does become important if you use transparent and/or translucent textures on some of your faces - transparent and translucent faces must be drawn last, otherwise other faces and lights will not be visible through them. You should therefore tell Blender which faces are transparent/translucent using the Alpha button in the Texture Face Editing panel (F9) in UV Face Select mode. The exporter will ensure that X-Plane draws these faces last.

But sometimes you need even more control over the drawing order - eg modelling a cockpit with a transparent HUD and (obviously) a transparent canopy; the HUD must be drawn after the canopy.

You can specify the relative order in which lamps, meshes etc should be drawn by assigning them to "Groups" on the Object and links panel (F7):

Objects that don't belong to a group are drawn first. The groups are sorted by alphabetical order, and then objects that belong to the first group are drawn next. … objects that belong to the last group are drawn last.

eg in the case of the cockpit with transparent canopy and HUD, we could put the canopy and HUD into separate groups named GroupA and GroupB respectively. Since A is before B in the alphabet, the canopy would be drawn before the HUD (and both would be drawn after the rest of the cockpit).

To add objects to a new group:

Select the meshes that you want to be drawn late. Press the Add to Group button on the Object and links panel (F7) (or press Ctrl-G). Choose ADD NEW. Give the new group a name.

Objects that belong to a group are highlighted in green instead of pink so that you can easily distinguish them.

v8/v9: Drawing group

You can specify when X-Plane should draw your scenery object relative to other scenery elements:

Add a Blender object of type "Empty" to your scene. Add a property to the Empty object named group_terrain, group_beaches, group_shoulders, group_taxiways, group_runways, group_markings, group_airports, group_roads, group_objects, group_light_objects or group_cars. Give the property a value between -1 and -5 to make X-Plane draw your object before this group, 0 to draw your object with this group, or between 1 and 5 to draw your object after this group.

eg to make X-Plane draw your object at the same time that it draws runway markings add a property named group_markings to an Empty object and give it the value 0.

v8/v9: Slung load weight

You can specify the weight of an object for use in X-Plane's physics engine if the object is being carried by a plane or helicopter:

Add a Blender object of type "Empty" to your scene. Add a property to the Empty object named slung_load_weight and specify the weight in pounds.

Optimising for X-Plane

v7: Strips

As well as basic 3- and 4-edged faces ("tri"s and "quad"s), X-Plane v7 objects support two compound types - "tri_fan" and "quad_strip". These are strips of two or more tris and quads that share common edges. Because the faces in these strips share common edges, X-Plane and the underlying OpenGL renderer have a third or a half as much work to do to render them compared to individual tris and quads. This gives higher frame rates.

In order for a pair of faces to be considered for inclusion in one of these strips, the following conditions need to be true:

The faces must be facing the same way. Each pair of faces must share a common edge (apart from the first and last face). Each shared edge must have the same texture co-ordinates in both faces.

In practice this means that the texture must be reversed in alternate faces in the strip. In the case of tris this can also be achieved by mapping a single area of the texture across all the tris, with the tip of the tris at the centre of the texture area. Use UVs → Copy & Paste from the UV/Image Editor window to automate the creation of strips.

These compound types aren't supported directly by Blender. However, the export script automatically tries to spot when it can use them.

v8, v9 & csl: The performance gains from using strips are more modest in X-Plane v8 and v9, and for CSL objects. These compound types aren't supported in v8/v9 objects (X-Plane v8 and v9 use more advanced techniques) and the only saving from using strips over some other UV mapping methods is in reduced "vertex count". It probably isn't worth going out of your way to look for opportunities of making strips.

Level Of Detail

X-Plane has borrowed the concept of "Level Of Detail" from 3D games. This works on the principle that when you're viewing an object from a large distance it can be displayed with reduced detail without you noticing the difference. By displaying distant objects with reduced detail we can simulate a more complex scene than would be possible if all objects were drawn at maximum detail.

Use Layers to draw scenery and CSLs (but not aircraft Cockpits or Misc Objects) with multiple Levels Of Detail. Objects in layers 1-3 are visible in X-Plane at the following distances:

Layer

Distance

1

< 1000m

2

1000-4000m

3

4000m-10000m

Changing the texture size

If you run out of space in your texture file then you can increase the size. The panel texture can by 1024×any size in X-Plane v8, and any size up to 2048×2048 in X-Plane v9. The height and width of a non-panel texture file must be a power of two eg 128, 256, 512, 1024 or 2048 (X-Plane v9 only), but it doesn't have to be square.

You should use the following procedure to ensure that your UV mappings and/or PlaneMaker instrument layouts are preserved:

Create a new, larger, texture in an image editor application. Panel texture: Paste the original texture into the lower left corner of the new texture. Non-panel texture: Paste the original texture into one of the corners of the new texture, or aligned on a multiple of the original texture's width and height. In Blender, in the UV/Image Editor window, select the original texture from the pop-up menu. Choose Image → Replace and fixup UV mapping… Select the new image and press Replace image. In the Fixup UV mapping dialog, press the button in the cluster of buttons that represents where you placed the original texture in the new texture.

You can also use this technique if you want to combine 3D models that use different texture files; paste all of the textures used by the 3D models into a single new texture file, then use Image → Replace and fixup UV mapping… on each of the original textures.

APPENDIX (C)

DATAREFS FOR ANIMATING OBJECTS IN X-PLANE

Flight Model: (sim/flightmodel2)

Facing the front of the plane= Left= [1] Centre= [0] Right= [2] or [0] =first object, [1] =second object, and so on.

Left Ruder:	sim/flightmodel2/wing/rudder1_deg[0]
Right Rudder:	sim/flightmodel2/wing/rudder1_deg[0]
Left Aileron:	sim/flightmodel2/wing/aileron1_deg[0]
Right Aileron:	sim/flightmodel2/wing/aileron1_deg[0]
Left Elevator	sim/flightmodel2/wing/elevator1_deg[0]
Right Elevator:	sim/flightmodel2/wing/elevator1_deg[0]
Flaps Left:	sim/flightmodel2/wing/flap1_deg[0]
Flaps Right:	sim/flightmodel2/wing/flap1_deg[1]
Slats:	sim/flightmodel2/controls/slat1_delopy_ratio[0]
Speed Brakes Left:	sim/flightmodel2/controls/speedbrake_ratio[0]
Speed Brakes Right:	sim/flightmodel2/controls/speedbrake_ratio[0]

Landing Gear:

Gear Deploy Ratio:	sim/flightmodel2/gear/deploy_ratio[0]
Gear Steering:	sim/flightmodel2/gear/tire_steer_actual_deg[0]
Gear Deflection:	sim/flightmodel2/gear/tire_vertical_deflection_mt[0]
Tire Rotation:	sim/flightmodel2/gear/tire_rotation_angle_deg[0]

Cockpit:

Throttle:	sim/flightmodel2/engines/throttle_used_ratio[0]
Engine:	sim/flightmodel2/engines/engine_rotation_angle_deg [0]
Canopy open/close:	sim/flightmodel2/misc/canopy_open_ration[0]

Cockpit Datarefs: (sim/cockpit2) Used for Manipulators:

Navigation Lights:	sim/cockpit2/switches/navigation_lights_on on value=1 off value=0
Strobe Lights:	sim/cockpit2/switches/strobe_lights_on
Taxi Lights:	sim/cockpit2/switches/taxi_lights_on
Battery:	sim/cockpit2/electrical/battery_on[0]
Igniter:	sim/cockpit2/engine/actuators/igniter_on[0]
Avionics:	sim/cockpit2/switches/avionics_power_on
Generator:	sim/cockpit2/electrical/generator_on[0]
Door open/close:	sim/cockpit2/switches/door_open[0]
Parking Brake:	sim/cockpit2/controls/parking_brake_ratio
Tail Hook:	sim/cockpit2/switches/tailhook_deploy
Camera:	sim/cockpit2/switches/camera_power_on

Cockpit Commands: Used for Manipulators:

APU:	sim/electrical/APU_start sim/electrical/APU_off
Inverter:	sim/electrical/inverter_on sim/electrical/inverter_off

Pitot Heat:	sim/ice/pitot_heat0_on sim/ice/pitot_heat0_off
Fuel On/Off:	sim/engines/engage_starters sim/starters/shut_down
Idle Hi/Low:	sim/engines/idle_hi_lo_toggle
Landing Lights:	sim/lights/landing_lights_on sim/lights/landing_lights_off
Landing Gear:	sim/flight_controls/landing_gear_down sim/flight_controls/landing_gear_up
Slider:	sim/operation/slider_01