Global Launch Vehicle Selector

A Senior Project

presented to

the Faculty of the Aerospace Engineering

California Polytechnic State University, San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree of

Bachelor of Science in Aerospace Engineering

by

Keisuke Suzuki

June, 2012

Global Launch Vehicle Selector

Keisuke Suzuki [1]
California Polytechnic State University, San Luis Obispo, CA, 93407

Launch vehicle selection is a crucial decision for any mission heading to space. When several launch vehicles are available as the potential carrier for a mission, this decision is no longer a simple matter. The Global Launch Vehicle Selector (GLVS) is a program which aims at supporting the customer with the launch vehicle selection. It allows the user to input parameters of the spacecraft such as payload mass and insertion orbit, and based on these values, outputs the most optimal launch vehicle as well as the launch site. GLVS is written in Javascript and C# within Unity 3 (Unity). Unity was chosen for its user friendly interface, ability to integrate 3d models, and its scripting environment to create a GUI. GLVS is published for use on either the PC or Mac.

Nomenclature

GLVS = global launch vehicle selector
GUI = graphical user interface

I. Introduction

Space based missions are still reliant on launch vehicles. However, the launch vehicle itself has advanced substantially in reliability, performance, and availability. The availability of launch vehicles, especially, has come a long way. More and more countries are starting to possess their own launch vehicle, while others are steadily advancing with their development of launch vehicles in the hopes of possessing one in the future and marketing them as business. Although the United States and Russia are the widely acknowledged leaders in aerospace, countries such as China are steadily increasing its presence. The diversity seen with the end users of the launch vehicles are changing as well as we see more international customers and academic institutions provided with the opportunity to send their payload into space[1].

While a launch vehicle is essential for a space mission, a customer may not be limited to selecting one from a domestic fleet of launch vehicles. If there is a cheaper option, a customer may purchase an international launch vehicle provider for launch. The same thing can be said for customers from countries which does not have its own domestic fleet of launch vehicles. Many satellites used by African or Asian nations are manufactured and launched abroad since they do not have the infrastructure to launch a spacecraft. One example is Nigeria's Nigcomsat 1R, a communication satellite which was manufactured and launched by China[2]. Pakistan is another country which relied on China's launch vehicle to send its satellite into space, while Vietnam is working with Japan to send its satellite into space[3,4]. Canada, having no launch vehicles of its own, sends its payload with launch vehicles from Europe or the United States[5]. The availability of launch vehicles should only increase for international satellite suppliers as more private companies make their way into the launch vehicle market demonstrated by companies such as Space X and Stratolaunch.

To better understand the fleet of launch vehicles available around the world, the Global Launch Vehicle Selector (GLVS) was created. The GLVS will serve two purposes for the user: a launch vehicle selection tool, and a database of the active launch vehicles. To help the process of selecting a launch vehicle, GLVS will provide a tool where users can input payload specifications such as payload mass, payload volume, insertion orbit etc. Based on the inputs, the tool will select all appropriate launch vehicles to launch the payload. As for the database, it will serve as a visual resource which can be used to aid students in learning more about launch vehicles.

[1] Undergraduate Student, Aerospace Engineering Department, San Luis Obispo, CA 93407.

II. Objective

The idea of GLVS was inspired from a presentation compiled by ESA which discussed about creating a Launch Vehicle Selector Tool[6]. The original idea looked at utilizing MATLAB to create a GUI where users could input their payload specifications. GLVS will incorporate the idea of this GUI tool, but would look to add more weight on accessibility by a wider range of users by having an interface not attached to MATLAB. A general overview of the user group which the GLVS expects to serve is shown in Fig.1. GLVS will serve as an open application where it will have enough functions to be a useful tool to select a launch vehicle, while at the same time, provide enough reference about launch vehicles for browsing purposes. The database will be created so that non-students or non-engineers can browse through it and gain a sense of what is flying into space.

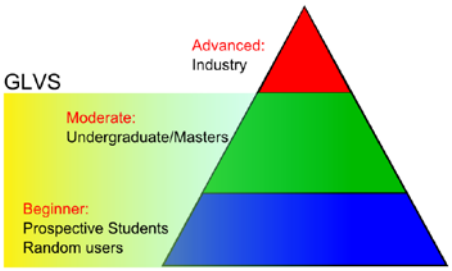


Figure 1. Target group for GLVS.

III. Approach

As stated earlier the GSLV will aim to serve two purposes. One will be a launch vehicle selection tool where a user can obtain feedback on which launch vehicle to use for his/her payload. Another is the database, or catalog of launch vehicles in which a user will find information on all active launch vehicles. The goal is to have an interface where these two functionalities will complement each other within the software. A simplified block diagram of the overall project can be seen in Fig. 2. For this project, only the functionality aspects of the GUI were worked on.

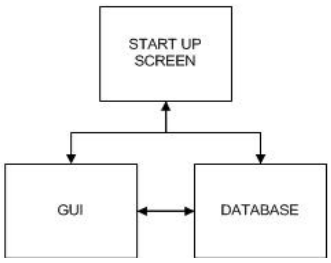


Figure 2. A general overview of GLVS's final design.

A. Platform/ Working Environment

In order to start on the project, a platform to work in was required. Many platforms provide functions to create a GUI which will be capable of making the launch vehicle selection tool GUI within GLVS. However, the final product would ideally support visuals and have rendering capabilities. This prevented MATLAB from being a possible candidate to work with. Nonetheless, the visual requirement helped in narrowing down the candidates for the work environment. Other platforms included Microsoft Visual Studio 2010 Express, and Adobe Dreamweaver CS5.5. These did not work out due to the fact that their applications branched out to so many areas; with the limited time to work on GLVS, the time needed to learn and locate the correct tools within these softwares made them an unfavorable choice. Time was another factor that needed to be considered. To overcome this, the working environment needed to have a predefined set of functions, or building blocks which simply needed to be placed in the right order rather than having to make everything from scratch. Hence, starting from scratch with a programming language such as C and C++ were also ruled out.

Fortunately, a game development tool called Unity 3 (Unity) showed great potential as a platform for creating GLVS. The strong point in Unity is the user interface. It comes with a layout in which one can work freely to obtain the desired looks and function. Another unique aspect is that it can import 3D models: this will help in making the database within GLVS. The deciding factor was that the basic Unity software is free. Although, there is a professional version which is otherwise, enough functions are available with the free version; therefore, it did not cause any significant problems.

B. Programming Languages

Unity supports three programming languages: Javascript, C#, and a dialect of Python named Boo. For this project, Javascript is used for coding. To be accurate, the Javascript incorporated in Unity is a modified version and is sometimes referred to as Unityscript; however, in this paper, it will be referred to as Javascript. The scripts are written in an external editor. There are several script editors that are compatible with Unity, and it is up to the creator's preference what to use. Notepad++ was used as the editor for this project.

C. Publishing

Unity is powerful in that it allows the user to publish his/her work on multiple platforms without any additional changes to the work itself; all it takes is selecting the platform one wishes his/her work to be published in. GLVS will be published so that it can be run on either a PC or Mac. Although not part of the project, Unity also supports web publishing which will allow for distribution of the project to a wider audience. Once GSLV has matured enough, this option can also be considered.

IV. GLVS Architecture

As mentioned earlier, the scope of this project will cover the selector tool only. Information here on will mainly be concerned with the selector tool. A general architecture of the whole GLVS is shown in Fig.3. Scripts were created using an external editor named Notepad++. There are multiple editors supported by Unity, and any of them will work fine for script editing. These scripts will determine how different elements on the screen should behave. The elements, or components, seen on the screen were all generated using ones provided within Unity: nothing was imported except for textures which used images downloaded from the web. A scene is basically the screen a user sees when using the GLVS. GLVS has more than 20 scenes and all of them can be navigated to using the navigation buttons located on each scene. The destination and the triggering of the navigation buttons are all controlled by the scripts.

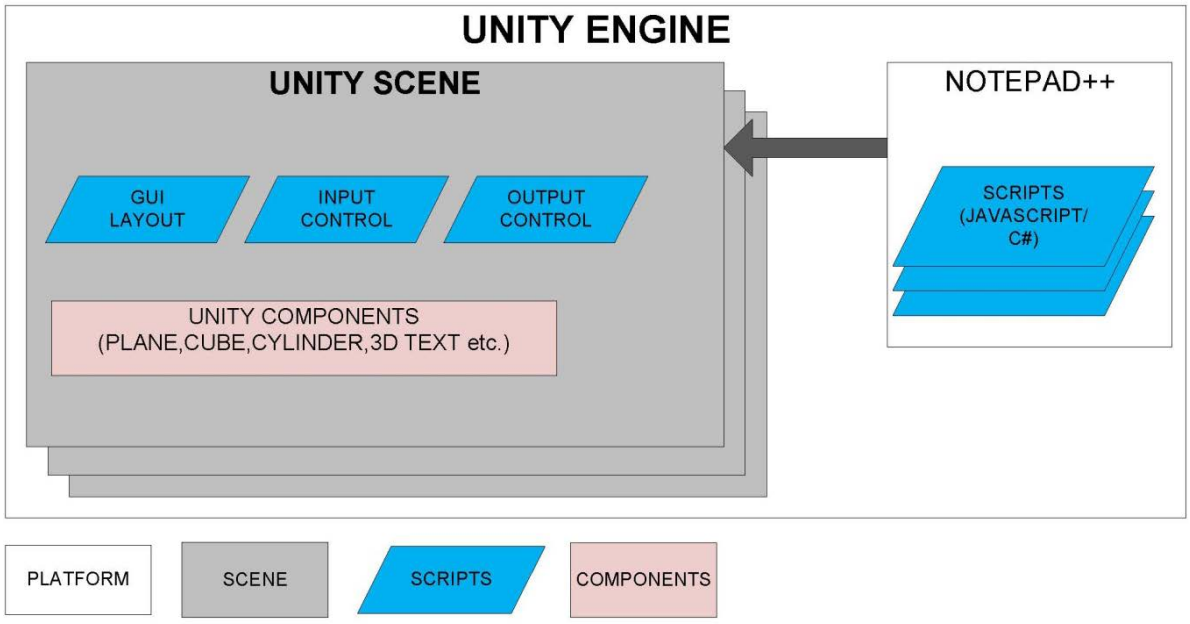


Figure 3. The overview of the GLVS architecture.

A. Inputs

User inputs should include the following:

1) Payload mass
2) Payload maximum cross-sectional diameter
3) Insertion orbit
4) Inclination

5) Launch site
6) Reliability

These inputs will be available for the user to specify, but at the same time, the user will have the option to leave some inputs blank. In order to obtain an output, however, the user will need to input at least the insertion orbit type. The GUI will ultimately be made so that it is flexible enough to accommodate any combination of inputs. Two types of inputting format will be available for user inputs. One will be a simple text area where the user can type any value or string. The other will be a pop down menu from which a user can pick a predefined parameter. A pop down menu will be utilized for inputting insertion orbit, launch sites, and use of nuclear powered components. Units for mass and dimensions are in metric units. English units are not considered for the current design. The list of launch sites is presented in Appendix A. Reliability number will be the success rate of the overall campaign in percentages. These reliability numbers are based off of the launch performance up to December 31, 2010[7].

B. Outputs

The outputs include the following list:

1) Launch vehicle
2) Launch site
3) Acceptable payload mass

The output will include all the launch vehicle that fits the user's need to carry the payload to space. The list of launch vehicles that was incorporated is shown under Appendix B (only active launch vehicles). The output will also consist of the payload mass. If the user has a mass value, the appropriate launch vehicles which can afford such mass will be outputted. In the case where the user does not have a mass input, the output will consist of the maximum launch mass for each of the appropriate launch vehicles.

The selection process will include a comparison chart, or a set of equations depending on the parameters inputted by the user. Derivation of the payload mass capacity available for different orbits is based off of equations generated by the curve fits of performance plots obtained from the user's manual guide[7] for each launch vehicle (also listed in Appendix C). Some errors are associated with the curve fit, and will be discussed later on. Other parameters such as the reliability and payload dimensions are solved by signs of equality or inequality.

C. Code Relations

All coding will be done within the Unity supported editor (Notepad++) using Javascript. Although Unity provides building blocks to create a GUI, scripts will be required to control its functions. In Fig.4, the general flow of the different scripts which orchestrates the launch vehicle selection processes is shown. Table 1 describes each of the variables shown in the figure.

V. Accuracy of Mass calculation

GLVS has some limitations with its ability to accurately determine the launch vehicle performances in different orbits. The first obstacle is presented by the curve fitting process. Since the best estimation for launch vehicle performance is provided as a plot, and not as a equation, curve fits were created to obtain an equation which may be incorporated within the script. Upon curve fitting, several points on a plot were manually chosen and recorded on Excel. Reading the values between grid lines on the plot inevitably lead to some inaccuracies. Although curve fits were created using Excel first, it was soon realized that the generated equations were not accurately representing the plots. Percent errors above 10% were observed when Excel's equations were imported into Unity. The reason was due to the first coefficient of the equation. When a curve fit is represented with a polynomial higher than the 3rd degree, the very first coefficient is given using one significant number, and thus, is rounded to the nearest unit. To avoid such occurrences, MATLAB was used utilizing its curve fit tool. Known as cftool, its curve fit equations allow for coefficients with multiple significant numbers. Since MATLAB excels in numerical operations, curve fits with a R-squared value of 1 was obtainable by having a polynomial of the 9th degree. However, the second obstacle showed itself when a high order polynomial was imported into Unity. Unity simply does not have the power to solve for such polynomials. For overwhelming numerical operations, the answer was simplified to infinity, whereas in MATLAB, the answers could be shown as numbers. Hence, equations could only be accurate as Unity's numerical solver's capability. This being said, all mass calculations based off of MATLAB's equations give answers with a percent error below 2.5% which is significantly below those obtained using Excel.

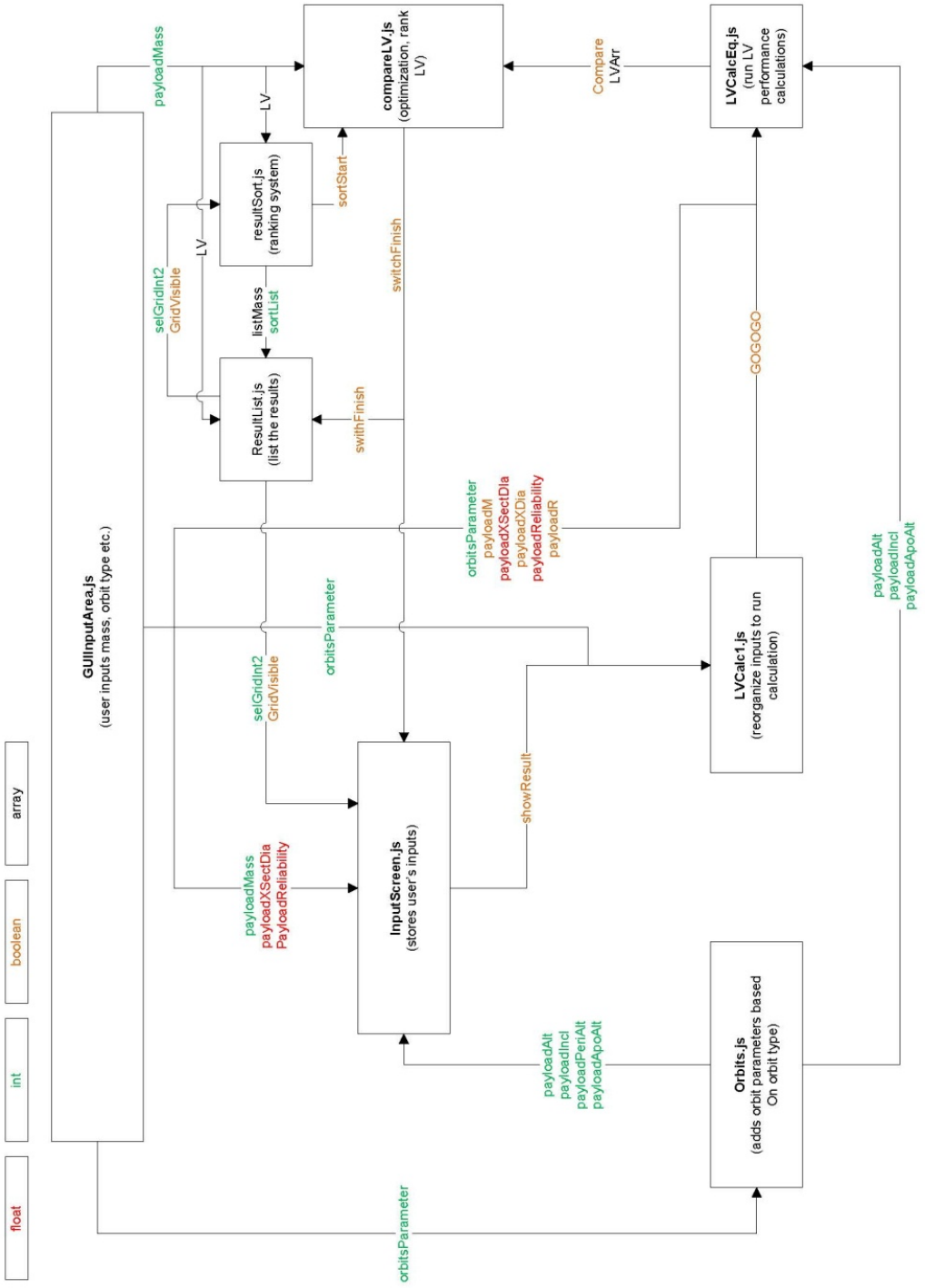


Figure 4. Flow of the static variables.

Table 1. The description of the static variables used within the scripts.

Script Name	Static Variables	What	Description	Connection
GUIInputArea.js	payloadMass	int	user input for payload mass	inputScreen.js; compareLV.js
	payloadM	boolean	identifies if user has a mass input	compareLV.js
	orbitsParameter	int	identifies user's insertion orbit types	Orbits.js; LVCalc1.js; LVCalcEq.js
	payloadXSectDia	float	user input for payload maximum cross sectional diameter	inputScreen.js; LVCalcEq.js
	payloadXDia	boolean	identifies if user has a diameter input	LVCalcEq.js
	payloadReliability	float	user input for LV reliability	inputScreen.js; LVCalcEq.js
	payloadR	boolean	identifies if user has a reliability input	LVCalcEq.js
Orbits.js	payloadAlt	float	user input for payload altitude	inputScreen.js; LVCalcEq.js
	payloadA	boolean	identifies if user has an altitude input	
	payloadIncl	float	user input for payload inclination	inputScreen.js; LVCalcEq.js
	payloadI	boolean	identifies if user has an inclination input	
	payloadPeriAlt	int	user input for payload perigee altitude	inputScreen.js;
	payloadPeriA	boolean	identifies if user has a perigee input	
	payloadApoAlt	int	user input for payload apogee altitude	inputScreen.js;LVCalcEq.js
	payloadApoA	boolean	identifies if user has an apogee input	
inputScreen.js	showResult	boolean	if true, starts calculation process	LVCalc1.js
LVCalc1.js	GOGOGO	boolean	if true, calculation process proceeds to step2	compareLV.js
LVCalcEq.js	compare	boolean	If true, starts compiling selected LV	compareLV.js; inputScreen.js
	LVArr	array	stores list of [LVname,mass, launch site]	compareLV.js
compareLV.js	switchFinish	boolean	if true, signals comparison has finished	inputScreen.js, ResultList.js
	LV	array	stores filtered LV name+mass	resultSort.js, ResultList.js
resultSort.js	listMass	array	array with LV ranked based on mass	ResultList.js
	sortStart	boolean	signal start of the sorting process	compareLV.js
	sortList	int	identify the parameter to rank	ResultList.js
ResultList.js	selGridInt2	int	identifies picked LV from grid list	inputScreen.js, resultSort.js
	gridVisible	boolean	start grid creation process	inputScreen.js, resultSort.js

VI. Current State

The current state of GLSV has the basic layout complete so that the user can switch screens using the buttons located on each of the screens. The GUI functions are in a mature state where user can input values and obtain an output. The selector tool is capable of outputting a launch vehicle, launchable mass, and launch site based on the payload mass, insertion orbit, payload cross sectional diameter, and launch vehicle reliability. The output can also be sorted by launchable mass in which the launch vehicles with the higher launch capacity will be shown at the top of the list. Although not as complete, two more approaches to the selection process is included on the GLVS. One is choosing a launch vehicle beforehand. This will help when a user knows a specific launch vehicle he/she wants to use. Upon selection, the user will obtain detailed specification of the launch vehicle to further assist the user with the payload design. The full specifications for each of the launch vehicles are yet to be included. The other approach narrows down the launch vehicle via a launch site. This will come in useful when launch location is known. This will also help identify what kind of launch vehicles launches from each site. The launch vehicles available at each site are referenced in the user's manual, but it is not available on the current GLVS. This will be one of the future works.

VII. Future Work

GLVS requires more work to near its ideal design where functionality is supported by visuals. 3D models of launch vehicles are not created at the time of this writing, and will be necessary as future work to achieve an intuitive aerospace tool. It should be included once work is started for the database side of the GLVS.

As for the functional aspects of the GLVS, there will always be space for improvements as that is the nature of computer programs. Specifically, improvements can be made in optimizing the indexing of the launch vehicles during internal calculations to speed up the program. The program experiences little lag as of now; however, since scripts were written while simultaneously learning the programming language there will definitely be space for improvements in terms of overall flow. Also, the performance of the selector tool is influenced by the amount of data accessible through a user's manual guide, textbooks, and online resources (keeping in mind that some are unreliable sources). Any additional information about the launch vehicle performances is always helpful in order to improve the GLVS.

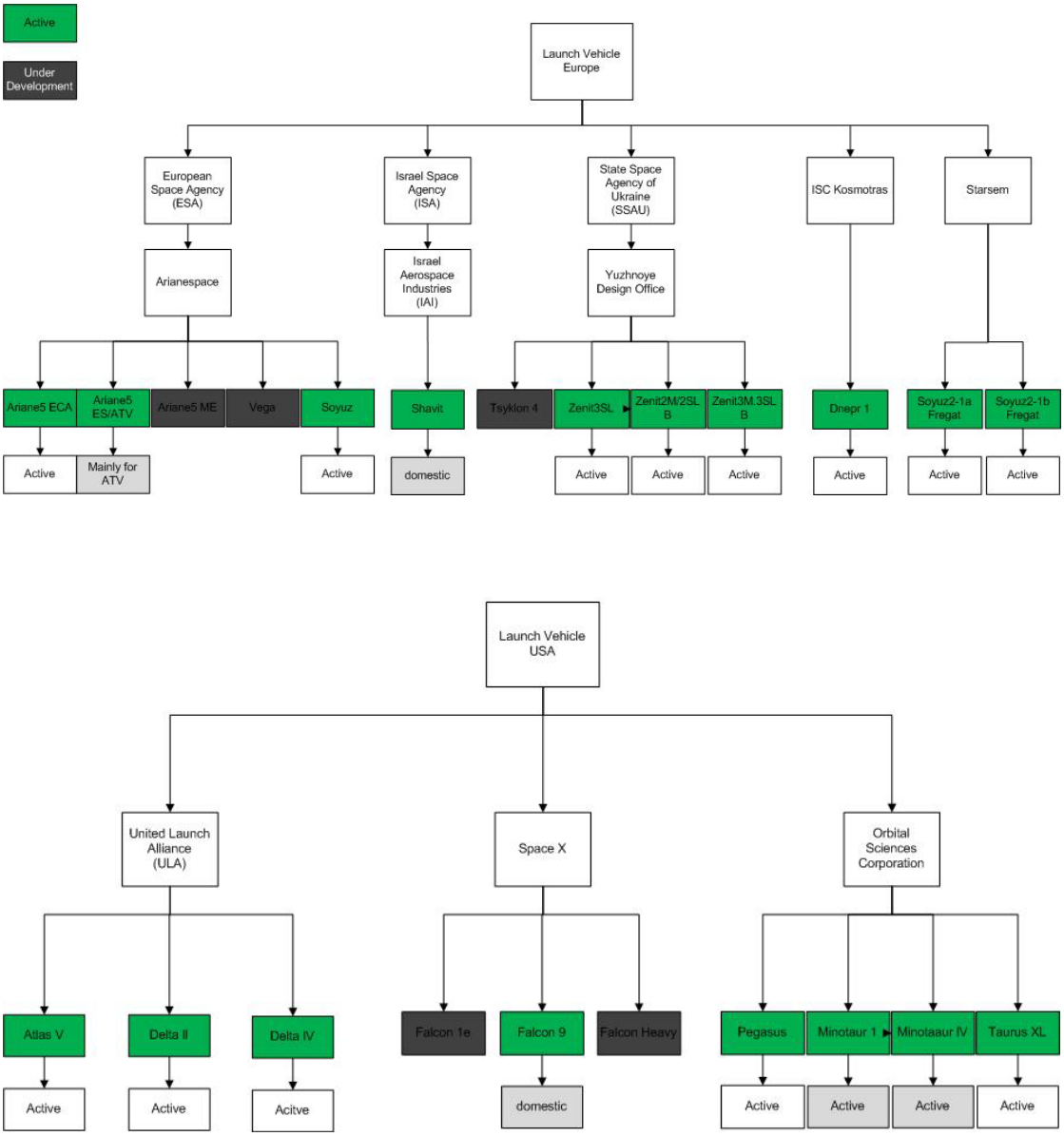
VIII. Conclusion

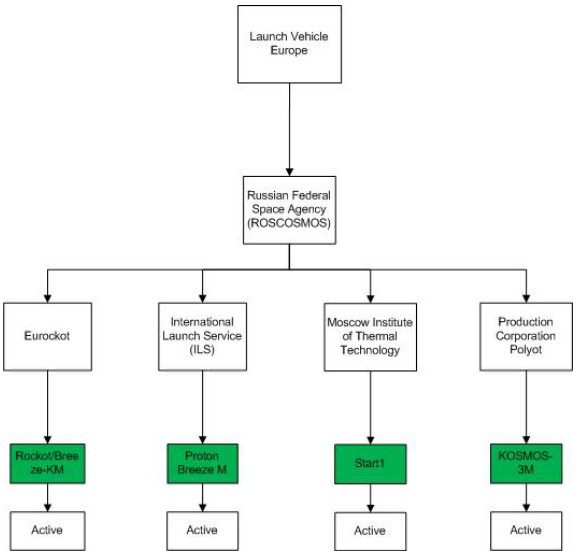
The GLVS has progressed far enough to allow a user interaction between it. A user can input a payload specification and obtain a launch vehicle successfully. The visual design of the program will be left for future work. So far, the potential of Unity as an engineering tool platform looks promising. The only huge obstacle stood in gathering data for launch vehicle performances. Availability of these pieces of information varied widely depending on the country of origin, and maturity of the launch vehicle.

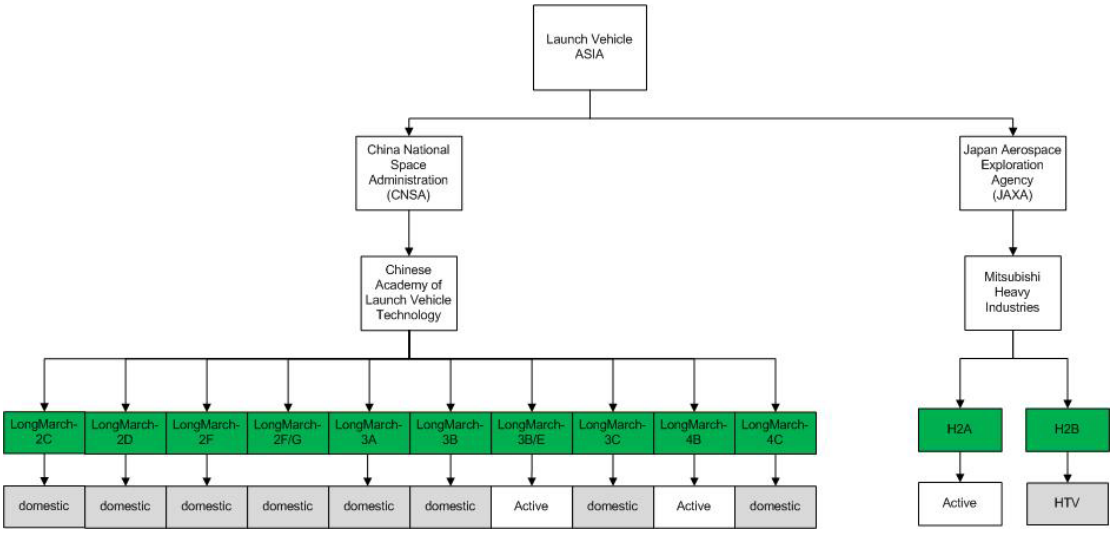
Appendix A: Available launch sites. *(SC = Space Center, FF = Flight Facility, AFS = Air Force Station, AFB = Air Force Base)*

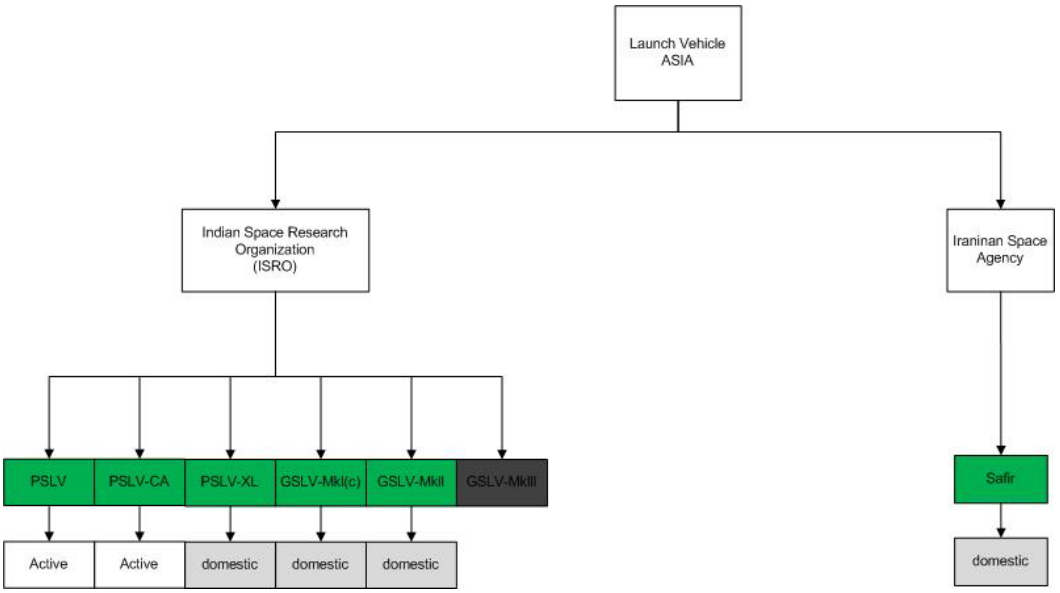
Country	Launch Sites
French Guiana	**Guiana Space Center**
Kazakhstan	**Baikonur Cosmodrome, Dombarovsky Cosmodrome**
Russia	**Kapustin Yar, Plesetsk Cosmodrome**
China	**Jiuquan SC, Taiyuan SC, Xichang SC**
Japan	**Tanegashima SC**
United States	**Wallops FF, Cape Canaveral AFS, Vandenberg AFB, Kodiak Launch Complex**
Republic of Marshall Islands	**Kwajalein Missile Range**
Iran	**Semnan Space and Missile Center**
Israel	**Palmachim AFB**
India	**Satish Dhawan SC**
(Pacific Ocean)	**Odyssey Platform**

Appendix B: The launch vehicles looked into for use in GLVS.




Launch Vehicle Europe
Russian Federal Space Agency (ROSCOSMOS)
Eurockot
International Launch Service (ILS)
Moscow Institute of Thermal Technology
Production Corporation Polyot
Rockot/Breeze-KM
Proton Breeze M
Start1
KOSMOS-3M
Active
Active
Active
Active


Launch Vehicle ASIA
China National Space Administration (CNSA)
Japan Aerospace Exploration Agency (JAXA)
Chinese Academy of Launch Vehicle Technology
Mitsubishi Heavy Industries
LongMarch-2C
LongMarch-2D
LongMarch-2F
LongMarch-2F/G
LongMarch-3A
LongMarch-3B
LongMarch-3B/E
LongMarch-3C
LongMarch-4B
LongMarch-4C
H2A
H2B
domestic
domestic
domestic
domestic
domestic
domestic
Active
domestic
Active
domestic
Active
HTV



APPENDIX C: List of User Manual Availability (alphabetical order). *For launch vehicles without a user's manual, the International Reference Guide to Space Launch Systems was used* [8].

Launch Vehicle	User Manual
Ariane 5	User's Manual Issue 5, Revision 1, July 2011
Atlas V	Atlas V Launch Services User's Guide, Revision 11, March 2010
Delta IV	Delta IV Payload Planners Guide, September 2007
Dnepr 1	Dnepr SLS User's Guide Issue 2, November 2001
Falcon 1	Falcon 1 Launch Vehicle Payload User's Guide Rev 7
Falcon 9	Falcon 9 Launch Vehicle Payload User's Guide Rev 1
H-II	International Reference Guide to Space Launch Systems
Kosmos	International Reference Guide to Space Launch Systems
Long March 2C	LM-2C User's Manual, Issue 1999
Long March 2D	None found
Long March 2F	None found
Long March 3A	LM-3A User's Manual, Issue 1996
Long March 3B	LM-3B User's Manual, Issue 1999
Long March 3C	LM-3C User's Manual, Issue 1998
Long March 4B	None found
Long March 4C	None found
Minotaur I	Minotaur I User's Guide, Release 2.1, January 2006
Minotaur IV	Minotaur IV User's Guide, Release 1.1, January 2006
Pegasus	Pegasus User's Guide, Release 7.0, April 2010
Proton	Proton Launch System Mission Planner's Guide, Revision 7, July 2009
PSLV&GSLV	International Reference Guide to Space Launch Systems
Rockot	Rockot User's Guide, Issue 5, Revision 0, August 2011
Safir	None found
Shavit	International Reference Guide to Space Launch Systems
Soyuz (Arianespace)	User's Manual Issue 2, Revision 0, March 2012
Soyuz	Soyuz User's Manual, Issue 3, Revision 0, April 2001
Start I	START-1 Users handbook Volume I, Issue 1, May 31 2002
Taurus	Taurus Launch System Payload User's Guide, Release 4.0, March 2006

VEGA	User's Manual Issue 3, Revision 0, March 2006
Zenit 3SL	User's Guide, Revision B, July 2000

Acknowledgments

The author would like to thank Mr. Daniel J. Wait for his time and assistance offered throughout the duration of this project. The author would also like to thank Dr. Kira Abercromby for her advice to help improve this project.

References

[1]Lafleur, Claude. "The Spacecrafts Encyclopedia." *Vous êtes Sur: Claudelafleur.qc.ca*. Web. 14 Mar. 2012. <http://claudelafleur.qc.ca/Spacecrafts-index.html>.

[2]Stephen, Clark. "Spaceflight Now | Breaking News | China, Nigeria Team up for Broadcasting Satellite Launch." *Spaceflight Now*. SPACEFLIGHT NOW, 19 Dec. 2011. Web. 14 Mar. 2012. <http://spaceflightnow.com/news/n1112/19longmarch/>.

[3]Stephen, Clark. "Spaceflight Now | Breaking News | Satellite for Pakistan Launched by Chinese Rocket." *Spaceflight Now*. SPACEFLIGHT NOW, 11 Aug. 2011. Web. 14 Mar. 2012. <http://www.spaceflightnow.com/news/n1108/11longmarch/>.

[4]Umezu, Paul K. "Japan, Vietnam Sign Deal for Two Radar Imaging Satellites." *Japan Vietnam Sign Deal for Two Radar Imaging Satellites*. Space News, 4 Nov. 2011. Web. 14 Mar. 2012. <http://www.spacenews.com/contracts/111104-japan-vietnam-deal-radar-sats.html>.

[5]"Satellites." *CSA*. Canadian Space Agency, 8 Dec. 2011. Web. 14 Mar. 2012. <http://www.asc-csa.gc.ca/eng/satellites/default.asp>.

[6]Blasco, Ana. "LVST: Launch Vehicle Selector Tool." *LVST: Launch Vehicle Selector Tool*. ESA. Web. 14 Mar. 2012. <http://trajectory.estec.esa.int/Astro/3rd-astro-workshop-presentations/The%20LVST%20launchers%20analysis%20tool%20and%20its%20use%20at%20ESA.pdf>.

[7,8] Isakowitz, Steven J., Joseph P. Hopkins, and Joshua B. Hopkins. *International Reference Guide to Space Launch Systems*. Reston, VA: American Institute of Aeronautics and Astronautics, 2004. Print.