Developing Cargo Loading Software for Navy and Marine Aircraft

Jeremy Ludwig, Bart Presnell, & Daniel Tuohy
Stottler Henke Associates, Inc.
San Mateo, CA 94402
ludwig, bpresnell, dtuohy @ stottlerhenke.com

Abstract—**Managing cargo loading for U.S. Navy and Marine Corps aircraft is a challenging task, requiring an understanding of elements such as aircraft limitations, aircraft center of gravity, cargo space dimensions, and tie-down procedures to name a few. These loading requirements are specified in each aircraft's lengthy Cargo Loading Guide (CLG). To address the problem of efficiently and effectively stowing cargo, the U.S. Navy has proposed the development of an Android app that assists aircrew in completing their loadmaster duties.**

This paper describes the Aircraft Cargo Evaluator app, which uses three specific capabilities to perform calculations and provide feedback to help achieve efficient and effective cargo loading. The first creates 3D models for novel cargo using Augmented Reality. The second allows the user to develop scenarios that include 3D models of aircraft, cargo, and tie-down patterns, and then analyzes the tie-downs according to CLG-defined rules. The third uses genetic algorithms to automatically search for and efficient and effective tie-down patterns for a scenario. The primary contribution of this work is summarizing how existing tools from augmented reality, computer games, and artificial intelligence were brought together to rapidly prototype an end-to-end solution in this challenging domain – and then following what happens as this research prototype takes the first steps towards the reality of operational use.

TABLE OF CONTENTS

1. INTRODUCTION

Managing cargo loading for U.S. Navy and Marine Corps aircraft is a challenging task, requiring an understanding of elements such as aircraft limitations, aircraft center of gravity, cargo space dimensions, and tie-down procedures to name a few. These elements differ across aircraft and are documented in lengthy Cargo Loading Guides (CLGs). One solution to these challenges is developing an app that runs on an Android tablet and assists aircrew in completing their loadmaster duties, helping to ensure that cargo is stowed efficiently and meets loading requirements specified in the CLGs.

This paper describes the Aircraft Cargo Evaluation (ACE) app, which performs calculations and provides feedback to help achieve efficient and effective cargo loading. The ACE app has three primary requirements. First, enable the development of a 3D model of cargo placement and tie-down patterns. Second, evaluate the safety of the placement and tie-downs based on the information in the CLGs. Third, generate a complete solution, or finish a partial solution, to a specified problem.

ACE includes three specific capabilities to achieve these requirements. The **AR Cargo Creator** creates 3D models of novel cargo using Augmented Reality (AR). The **3D Editor** allows the user to develop a 3D model of cargo and tie-down patterns, and then analyzes the tie-downs, weight, and balance according to CLG-defined rules. The 3D Editor includes representations of multiple aircraft platforms. The **GA Generator** uses genetic algorithms (GAs) to automatically generate efficient and effective tie-down patterns for a 3D model.

The following is an illustrative use case that ties these capabilities together. Using ACE, a loadmaster or crew chief uses the 3D Editor to quickly create a 3D model that includes the aircraft configuration and cargo to be placed. In the less frequent case where one of the cargo items has not already been modeled, they will use the AR Cargo Creator to create a new cargo model. At this point, the user could either tie-down the cargo manually and press the 'Analyze' button to ensure the correctness of the solution or press the 'Generate' button to invoke the GA Generator and begin the search for a valid solution.

There are two main benefits to using ACE, as opposed to the manual calculations that currently occur. First, there is a large number of constraints and issues that must be considered when securing a load of cargo. Doing these calculations manually, and ensuring none are missed, is a time-consuming task. By automating these calculations, ACE will help loadmasters and crew chiefs to complete their work significantly faster than the current manual process. Second, the current system of manual calculation does not provide any feedback to inform the loadmaster of a possible mistake that might lead to a catastrophic event — either on the current flight or on a future flight if the mistake is repeated. ACE provides specific, actionable feedback that helps the user correctly secure cargo in the current flight and provides loadmaster training through positive examples.

The primary contribution of this paper is summarizing prior work that describes how existing tools from augmented reality, computer games, and artificial intelligence were brought together to rapidly prototype an end-to-end solution in this challenging domain [1] – and then following what happens as this research prototype moves towards the reality of operational use. The Related Work section briefly describes the tools used in ACE as well as an entry point for research on optimizing weight and balance of aircraft cargo. Following that, the Methods and Results sections provide an overview of the software, describe the initial evaluation process, and highlights changes based on end user feedback. The Conclusion summarizes progress on ACE to date and outlines future work.

2. Related Work

Augmented reality (AR) aims to provide an interactive experience the combines both real and virtual worlds [2]. One challenge of augmented reality is creating accurate 3D virtual models of real-world objects to interact with. The **AR Cargo Creator** component is tasked with just this problem - creating 3D models of novel real-world cargo. To do this, the AR Cargo Creator leverages the Unity Augmented Reality (AR) toolkit [3], which supports the development of immersive applications that interact with virtual and real-world objects. Specifically, we use the various features like plane tracking that are designed to capture 3D models. By using this toolkit ACE builds on the massive amount of research that has gone into creating 3D models of real-word objects for AR systems. Snap2Cad [4] is a good example of ongoing research in this area related to ACE. The Snap2Cad system uses information from an Android camera and the Google ARCore toolkit [5] to retrieve the most similar object from a library of 3D models, scale the model, and then visualize the combined scheme. ACE will benefit as research such as this is incorporated in the AR toolkits.

Unity [6] provides a solid foundation for the **3D Editor**. Unity is an extremely popular real-time 3D development platform with extensive support for mobile devices. Unity supports the development of the features required by the user to place and restrain virtual cargo in a simulated aircraft on an Android tablet.

Genetic Algorithms (GAs) are a search technique inspired by the evolutionary process in biology [7]. The basic GA process is:

1. An initial population of solutions is created.
2. Offspring solutions are created through the process of combining two solutions through crossover and changing a solution through mutation.
3. A fitness function determines which offspring survive to create more offspring.

This process is continued until some threshold is reached such as number of generations or fitness threshold. ACE uses the open-source GeneticSharp [8] library as a foundation for the **GA Generator** to generate efficient and effective tie-down patterns.

There is significant related work in properly restraining cargo as well as analyzing and optimizing aircraft weight and balance. For example, the U.S. Air Force's Air Transportability Test Loading Activity office maintains formulas and tools used by all federal agencies in the areas of restraint, weight, and balance [9]. However, we did not find any related work developing intuitive, tablet-based, software for analyzing cargo restraint, weight, and balance as found in ACE.

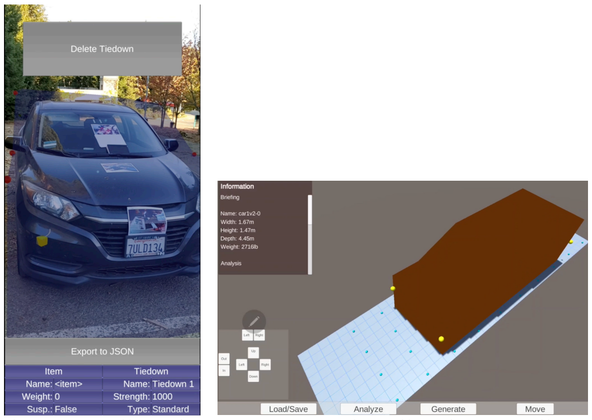
As an example of work optimizing weight and balance, [10] describes a mixed integer programming model that maximizes payload and minimizes center of gravity deviation. The authors provide an in-depth review of related work on air cargo weight and balance optimization. They also note that while there are existing visual software tools that calculate aircraft weight and balance many airlines still perform this task manually. While ACE does perform weight and balance calculations to ensure they are within guidelines, it does not try to optimize cargo placement and instead relies on the user to place the cargo.

3. Methods

This section first summarizes the three main components included in the initial ACE proof of concept [1] (previously name AutoLoader) and then describes the evaluation process that was carried out. The three components are the User Interface, Computational Engine, and 3D Models.

User Interface

The **AR Cargo Creator** functionality leverages the Unity AR Toolkit to allow the loadmaster to interactively create a 3D model from a real-world object, add tie-down points to the model, and to import the model into the 3D editor app as shown in Figure 1. Outdoor testing of the AR Cargo Creator on a small SUV demonstrated reasonable accuracy with picture-based plane detection, where known images are placed on specific planes. These known images are printed onto magnets and labelled with their plane (e.g., right side), making them relatively straightforward to use. The system worked under a wide range of angles and varying light, with the limiting factor that the picture must fill about 50% of the frame.



Item	Tiedown
Name: <item>	Name: Tiedown 1
Weight: 0	Strength: 1000
Susp.: False	Type: Standard

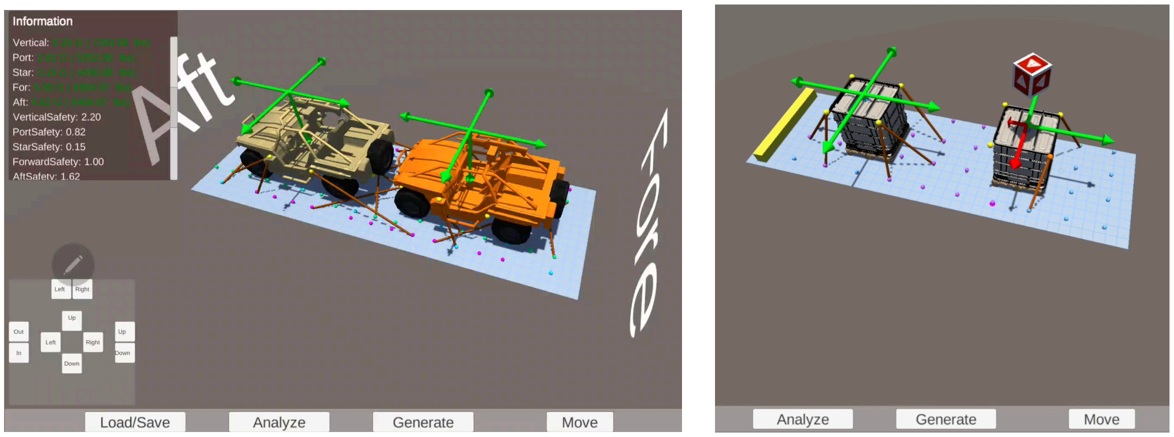
```
Information
Briefing

Name: car1v2-0
Width: 1.67m
Height: 1.47m
Depth: 4.45m
Weight: 2716lb

Analysis
```

Figure 1. Creating a Model of a Car in AR Cargo Creator (left), Importing the Model into the 3D Editor (right).

The **3D Editor** relies on the Unity Engine to visualize cargo placement and tie-down patterns. The 3D Editor is also responsible for displaying the results of the Validator, which include the restraint in each direction and that restraint and other cargo loading rules are followed. Figure 2 (left) shows an example of sufficiently restrained cargo in a heavy lift helicopter. The green arrows indicate sufficient restraint in all directions. The lack of a warning icon above an object indicates that all restraint rules are followed. Figure 2 (right) illustrates an incorrectly restrained item in another aircraft, where the cargo box on the right has both insufficient restraint and rule violations. The cargo box on the left is properly tied down. Finally, the yellow blocks on the far left indicate unavailable/unusable tie-down locations.

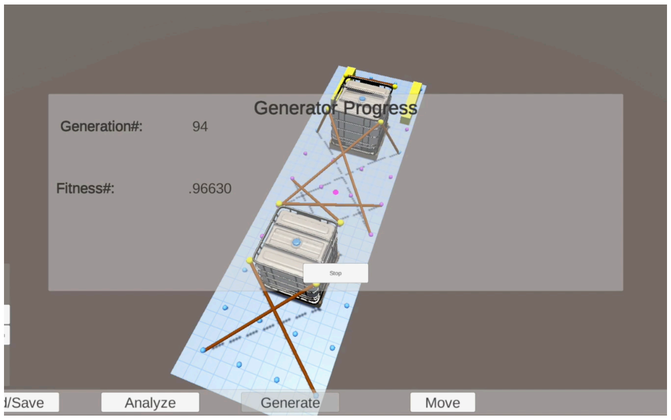
Computational Engine

The computational engine includes both the cargo Validator and **GA Generator**. Once the cargo is placed and tied-down in the 3D Editor, the Validator analyzes the cargo restraint provided by tie-downs and the cargo weight and balance, utilizing the formulas outlined in the CLGs. Further, the editor validates that specific restraint rules are followed such as: do not use blocked tie-down points; if a cargo object has suspension, then 50% of the restraint must be above the suspension; the fore/aft and port/starboard restraints must be balanced; and mismatched restraints (e.g., chains and straps) should not be used.



```
Information
Vertical: 5.20 G ( 7284.58  lbs)
Port: 3.82 G ( 5352.95  lbs)
Star: 3.15 G ( 4406.88  lbs)
For: 5.00 G ( 6994.97  lbs)
Aft: 4.62 G ( 6464.87  lbs)
VerticalSafety: 2.20
PortSafety: 0.82
StarSafety: 0.15
ForwardSafety: 1.00
AftSafety: 1.62
```

Figure 2. Sufficiently restrained (left) and insufficiently restrained (right) cargo.



```
Generation#:    94
Fitness#:       .96630
```

Figure 3. GA Generator in progress.

Within the 3D Editor app, the 'Generate' button launches the **GA Generator**. This capability uses a genetic algorithm to automatically search for an efficient and effective tie-down pattern for a 3D model, as shown in Figure 3. The GA is implemented as a set of Unity scripts that extend and heavily modify the GeneticSharp [8] library . We chose to implement the GA natively inside Unity in order to exploit useful computations provided by Unity's 3D engine. In the GA representation, each chromosome is a solution. The chromosome is made up of an ordered list of genes, where each gene represents a pair of mirrored restraints between cargo and the platform. Crossover is performed at the level of entire pieces of cargo. That is, for each piece of cargo, all of the genes which encode the restraints for that piece are taken from a single parent. Likewise, each mutation operation is performed on pairs of restraints on a piece of cargo: restraints are added as pairs, deleted as pairs, and incrementally altered as pairs, with mirroring enforced at every step. The fitness function drives the search over time by favoring solutions where the required restraint is met in all directions, with fewer rule violations, with fewer and shorter straps, and encouraging crisscross restraints. The GA fitness function is structured so that it can be easily re-parameterized to weight each of these component factors differently. This supports searching for a variety of solutions, based on which criteria are weighted higher.

Models

A set of three aircraft platform models were developed for the prototype app based on the information in the CLGs. Additionally, a handful of representative pre-existing 3D models were used for the cargo items in the proof of concept.

Evaluation

Two informal evaluation events were carried out, where personnel from each of the three platforms participated in each event. The first event focused on understanding the issues and concerns of each individual platform. To foster this we carried out separate meetings over two days, discussing current operations and pain points as well as eliciting feedback by demonstrating the three primary ACE software capabilities. During this evaluation event, the AR Creator received the most attention. For the second evaluation event we gathered Crew Chiefs and Loadmasters from each of the three platforms, along with the Cargo Team and various program leads, into a single room. This evaluation event focused on hands-on evaluation of the 3D Editor and GA Generator on several Android tablets. The attendees enthusiastically participated in a dynamic, cross-platform, evaluation of the current app and discussion of what features should be tackled in the future.

4. Results and Discussion

This section summarizes the results from the two evaluation events and discusses the impact of these results on the ongoing software development.

First Evaluation Event

The results of the first evaluation event fall into one of several broad categories: use cases, fixes & features of each of the three main app capabilities, and general app features and concerns. The first result of this evaluation event was to document a set of use cases for ACE in each of the three platforms, focusing on the types of problems Loadmasters and Crew Chiefs typically encounter. They can be summarized as standard cargo items, a mix of standard cargo items, and unique cargo items. ACE focuses on the first two. As examples, standard items might include a number of pre-restrained 463-L pallets or a single vehicle, while a mix of standard items might include personnel, a vehicle, and some number of non-463-L pallets.

The second result focused on fixes and features in the three main app areas: AR Cargo Creator, 3D Editor, and GA Generator. The feedback for the 3D Editor generally took the form of incremental improvements. For example, making it easier to deploy seats, adding buttons to flip to specific viewpoints quickly, adding labels to aid placing the cargo, the ability to customize the cargo weight and dimensions, and performing weight and balance calculations to name a few. Surprisingly to us, the one item that seemed to be most annoying to the reviewers was the use of the metric system. Similarly, feedback for the GA Generator was iterative as well and aimed at producing the types of tie-down patterns that people would make. This includes adding tie-downs in pairs and making better use of crisscross patterns. The most interesting feature request was to provide a range of solutions according to different optimization functions, so that the user can select the solution most appropriate for the current conditions. For example, if the aircraft is short on straps it might be better to select a pattern with a bigger footprint that uses fewer straps. Feedback on the AR Cargo Creator was a little more transformative. In short, users found the custom magnets used for image-based plane detection to be unworkable in the field. Other avenues were discussed, such as using disposable stickers rather than magnets, using less-accurate unaided plane detection combined with measurements made with a tape measure, or to include pre-existing templates that could be customized with a set of measurements (removing the need for any augmented reality). The main takeaway is that creating 3D models of unique cargo with a tablet in an operational setting is a challenging problem and we need to pursue new avenues of research to address this.

The general app feedback focused on the Cargo Loading Guides (CLGs). First, reviewers would like to see the CLG publications included in the tablet. Second, reviewers suggested the app should walk the user through the relevant portions of the CLG. For example, the user would select and customize an item and then the app would help the user verify it will fit and can be loaded with the ramp down. Next the app would walk the user through any pre-loading requirements such as shoring or possible hazmat issues. After that the app would tell the user where to place the item and then provide step by step instructions for restraining it. The result is essentially a CLG customized specifically to the cargo being loaded.

The feedback from the first evaluation event was extremely valuable in helping guide software development. First and foremost, the event demonstrated that the technical proof on concept for the 3D Editor and GA Generator was sound. The use cases and incremental improvements were put directly into the issue list as future work in these two components. The feedback on the AR Cargo Creator, combined with the lower priority of handling unique cargo items, moved the software in the direction of customizing existing models instead of using the camera to capture new models (for now). Finally, the concept of a walk through 'wizard' has been captured as future work. Once the app can perform all of the individual steps required for a piece of cargo (e.g., projecting fit, determining a location, creating a tie down pattern), then we can begin to link all of the steps together to create the wizard.

Second Evaluation Event

In the intervening time between the first and second evaluation events, software development focused on incorporating user feedback and improving the quality of results found by the GA Generator. The 3D Editor remained largely the same, except for removing the AR Cargo Creator button so it would not be inadvertently accessed. While both the GA Generator and 3D Editor were demonstrated, the bulk of the feedback provided was aimed at the 3D Editor. The feedback falls into three broad categories: usability, platforms, and process integration.

The reviewers spent a lot of time identifying usability issues and developing more streamlined alternatives. The consistent message was to make the user interface and user experience as simple as possible while still supporting all of the necessary features. For example, not having undo/redo or clunky system for manually applying straps in a research proof of concept was perfectly acceptable – and not acceptable in a system moving towards operational use. Similarly, engineering outputs on the genetic algorithm fitness function are not suitable for end users. Instead, users need a clear understanding what the GA is looking for in a solution and if the found solution meets minimum requirements. Most of the items they identified were fairly straightforward to turn into issues to be addressed in future releases. Some identified issues do not have a straightforward fix, such as how to model multiple items in a big pile and then wrapping straps over it. These types of issues are being tracked for investigation in later versions.

Having representatives from three different platforms in the same room allowed us to quickly identify a variety of platform-specific issues and design solutions that would work across platforms. Platforms differ in their use cases, possible aircraft configurations, and their restraining criteria. For example, look at the relatively common scenario of carrying both personnel and cargo. If you have personnel and cargo which, do you place first? Do you try and put personnel forward or aft of the cargo? The answer is, not surprisingly, 'it depends'. The solution is to design the software such that the user can either directly select which seats are occupied and provide preferences for automated seat selection. Similarly, each platform has several specific configuration items that would affect weight, balance, and available cargo space. The user should be able to easily select which are in the current aircraft. Finally, each aircraft has its own set of rules detailed in the Cargo Loading Guide and it is imperative to use the set of rules matching the platform.

Feedback on process integration focused on how this app would fit into the existing, highly regimented, processes for verifying cargo before takeoff. For example, pilots and aircrew are already documenting weight and balance

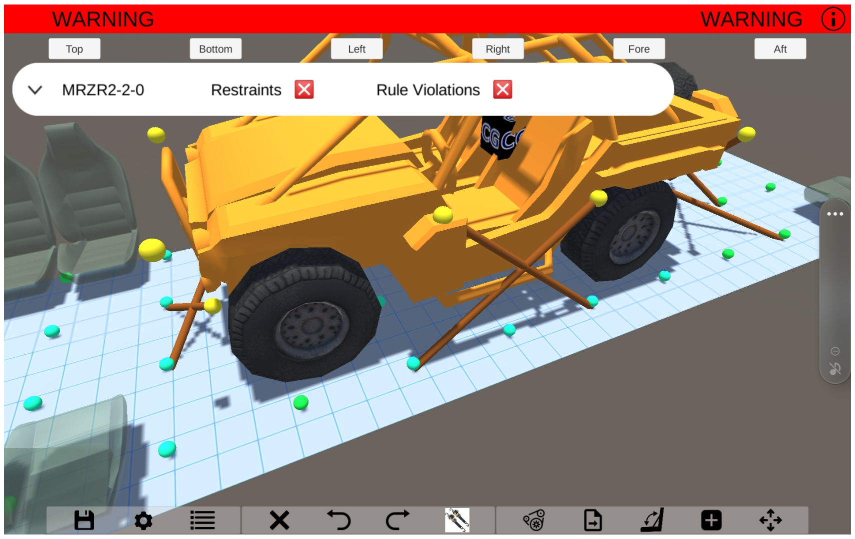


Figure 4. Updated user interface based on feedback from evaluation events.

calculations. Reviewers were interested in knowing how the results of this app could be included in the existing process, saving them time and energy. Also, while reviewers stressed that it was important to have as simple and intuitive user interface as possible, for verification and validation details on the underlying calculations will be needed. The result of this is to include an ‘advanced’ series of displays where the calculations can be presented and downloaded.

Discussion

An in-progress screenshot of the app user interface is shown in Figure 4. This version addresses many of the 3D Editor and GA Generator issues discovered during the evaluation process. For example, replacing yellow exclusion boxes with deployable seats, providing a set of view buttons along the top, and streamlining the available functions in the toolbar along the bottom such as undo/redo. While the current version does not yet address all the identified issues, it demonstrates significant progress from the proof of concept towards a minimum viable product.

5. CONCLUSION

Managing cargo loading for U.S. Navy and Marine Corps aircraft is a challenging task, requiring an understanding of complex requirements that differ across aircraft and are documented in lengthy Cargo Loading Guides (CLGs). The primary objective of the ACE system described in this paper is to assist aircrew in completing their loadmaster duties — helping to ensure that cargo is stowed efficiently and meets loading requirements specified in the CLGs.

With the **AR Cargo Creator**, we demonstrated the capability to develop an initial model using augmented reality to recognize planes, to manually refine the model and add tie-down points, and to import the created model into the 3D Editor. With the **3D Editor**, we demonstrated placing and tying down cargo in multiple aircraft. The engine quantified the restraint applied to cargo in each direction, validated that restraint rules were followed, and the editor communicated this information to the user. With the **GA Generator**, we demonstrated the ability to use genetic algorithms to efficiently search for a tie-down solution. The fitness function improved and refined results over time by ensuring the required restraint was met in all directions, minimizing the number of rule violations, preferring fewer and shorter straps, and encouraging symmetry. The proof-of-concept prototypes successfully demonstrate the technical feasibility of key aspects of ACE.

While the protype results successfully demonstrate the technical feasibility of key aspects of ACE, the evaluation events highlighted what is needed to move from the prototype to a minimum viable product. This work includes improving the usability of the tool and adding features based on end-user use cases. While technically feasible, the AR Cargo Creator concept needs to be replaced with something that is more practical for operational use. Additionally, the app needs to be updated to better support the needs of each of the three individual platforms and their user communities. Finally, looking towards the future the app needs to prepare for rigorous verification and validation and eventual integration with existing air cargo processes.

ACKNOWLEDGEMENTS

This material is based upon work supported by the United States Navy under Contract No. N68335-20-C-0538. The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the United States Navy.

REFERENCES

[1] J. Ludwig and B. Presnell, “Autoloader: Cargo Handling Software for Navy and Marine Aircraft,” presented at the IEEE Aerospace Conference 2022, 2022.

[2] “Augmented reality,” *Wikipedia*. Dec. 30, 2021. Accessed: Dec. 30, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Augmented_reality&oldid=1062792894

[3] U. Technologies, “Augmented Reality Development Software | AR Engine for Apps | Unity.” https://unity.com/unity/features/ar (accessed Nov. 11, 2020).

[4] A. Manni, D. Oriti, A. Sanna, F. De Pace, and F. Manuri, “Snap2cad: 3D indoor environment reconstruction for AR/VR applications using a smartphone device,” *Computers & Graphics*, vol. 100, pp. 116–124, Nov. 2021, doi: 10.1016/j.cag.2021.07.014.

[5] “Build new augmented reality experiences that seamlessly blend the digital and physical worlds | ARCore,” *Google Developers*. https://developers.google.com/ar (accessed Dec. 30, 2021).

[6] U. Technologies, “Unity Real-Time Development Platform | 3D, 2D VR & AR Engine.” https://unity.com/ (accessed Nov. 11, 2020).

[7] “Genetic algorithm,” *Wikipedia*. Nov. 11, 2020. Accessed: Nov. 18, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Genetic_algorithm&oldid=988134173

[8] “GitHub - giacomelli/GeneticSharp: GeneticSharp is a fast, extensible, multi-platform and multithreading C# Genetic Algorithm library that simplifies the development of applications using Genetic Algorithms (GAs).” https://github.com/giacomelli/GeneticSharp (accessed Nov. 11, 2020).

[9] “ATTLA ensures aircraft cargo gets to destination safely,” *Wright-Patterson AFB*. http://www.wpafb.af.mil/News/Article-Display/Article/818869/attla-ensures-aircraft-cargo-gets-to-destination-safely (accessed Jul. 08, 2022).

[10] X. Zhao, Y. Yuan, Y. Dong, and R. Zhao, “Optimization approach to the aircraft weight and balance problem with the centre of gravity envelope constraints,” *IET Intelligent Transport Systems*, vol. 15, no. 10, pp. 1269–1286, 2021, doi: 10.1049/itr2.12096.