Developing a Real-Time Weather Simulator

Noah LeFrancois[a], **Filipe Rodrigues**[a], **and Hong Guo, Dr.**[a]

[a]Department of Physics, McGill University, 3600 Rue Université, Montréal, QC, H3A 2T8, Canada

This manuscript was compiled on April 17, 2020

This project aims to construct a framework for visualizing scientific data in a virtual reality environment using the Unity game engine. For this purpose, we seek to develop a real-time weather simulator, capable of presenting in virtual reality the dynamic global flow of atmospheres on various planetary terrains. As an initial proof-of-concept investigation and basis for further work, this report and that of the second author develop a pipeline for mesh generation, fluid simulation, and data visualization by integrating the various software necessary for a working model. This report will focus on the simulation aspect of the project, while the visualization aspect will be addressed in the second author's report. Both reports will address aspects of the mesh generation process. We first generate the desired terrain and produce a mesh representing this geometry from the planet surface to the top of the atmosphere. The mesh file is used to set up a fluid simulation solving the Euler equations, discretized by the finite volume method and a second-order dual time-stepping method. Simple boundary conditions are given, and a highly simplified model of atmospheric dynamics involving convection in spherical coordinates will be used to maintain computational speed. A number of recommendations are made for the development of more realistic boundary and initial conditions as well as improved time-stepping methods; these will provide directions for future extensions to this project.

Science Education Technology | Weather Simulator | Compressible Euler Equations | Finite Volume Method

With the rapid increase in computer power and huge improvement in hardware/software technologies in recent years, now is an opportune time to investigate and produce a real-time weather simulator of reasonable quality. At the same time, the use of virtual reality (VR) for scientific data visualization is an emerging field which has exciting potential in a number of areas including producing engaging educational materials.

The combination of these two developments offers an opportunity to demonstrate the viability of VR as a medium for visualizing scientific data in an accessible and engaging manner. This medium has the potential to illustrate physical phenomena in a very different fashion than the traditional graphical method used in scientific papers, which is accessible only to a small group of highly educated individuals. A weather simulator capable of rendering atmospheric data in real-time using this technology has been identified for an initial demonstration due to the widespread importance of weather visualization for public audiences as well as the extensive literature available on weather simulation.

Atmospheric simulation may be divided into four aspects: dynamic global flow simulation, scattering & absorption in the atmosphere, local weather simulation for clouds & precipitation, and weather visualization. These aspects are all important and interdependent; achieving them in real-time is a lofty long-term goal. In this paper we will focus on the first aspect, creating an efficient simulation of global atmospheric flows. F.R.'s project focuses on the final aspect, weather visualization. Working with the Unity game engine, he will be developing a method of visualizing atmospheric flows in a VR environment. Together these projects will establish a framework upon which to build an effective real-time simulation and visualization of scientific data.

The study of computational fluid dynamics (CFD) allows us to better understand the behaviour of fluids in motion, and has "rapidly been adopted as a household methodology in solving complex problems in modern engineering practice." (1) Based upon the fields of fluid mechanics and heat transfer, its applications range from modelling new wing designs in aerospace engineering to quark-gluon plasmas in nuclear physics. As a result of its widespread usage, there exists a wide array of CFD software available to scientists today; many of these are open-source and readily available for anyone to use in their work. We will make use of such CFD software to simulate the dynamics of a planetary atmosphere in this project.

In our model of global atmospheric flow, the compressible Euler equations will be solved. These are a simplification of the Navier-Stokes equations, which govern fluid mechanics. In this approximation, the fluid is treated as having no heat conduction or viscosity. This approximation increases simulation speed and has been demonstrated as a viable method for global atmospheric modelling. (2)

A major advantage of this choice is that we can solve the Euler equations with relatively large grid size, allowing for large time steps to move us towards real-time speed.

Studies of atmospheric air flow have shown that this phenomenon is dominated by convection. (3) As a result, the finite volume method (FVM) has been identified as an effective

Significance Statement

Traditional graphical depictions of scientific data often require significant field-specific knowledge to understand and are inaccessible to the vast majority of the public. This project aims to produce a visualization of weather data with which users can intuitively interact in virtual reality. As VR technology has shown rapid improvement in both performance and affordability in the past decade, this visualization could demonstrate the viability of a much more accessible and engaging medium for displaying scientific data. For the purpose of this demonstration geared towards science education, we seek to build a simplified weather simulator which is fast enough to be used in real-time demonstrations of atmospheric flow.

N.L. was responsible for simulation set-up, numerical model selection, and writing of this report; F.R. was responsible for terrain geometry generation in Unity and post-processing of simulation data for visualization; both contributed equally to the exporting of geometry mesh from Unity into simulation.

[1]To whom correspondence should be addressed. E-mail: noah.lefrancois@mail.mcgill.ca

method for modelling it. (4)

The finite volume method discretizes the integral form of a conservation law directly in physical space by dividing the domain into control volumes, or cells, and approximating the conservation law on each volume. First employed for the solution of the two-dimensional time-dependent Euler equations in the early 1970s, the FVM is now the method of choice for most industrial CFD codes as it can accommodate any type of grid. This is because it works with control volumes instead of grid intersection points. (1)

Producing a functioning atmospheric solver requires the coordination of numerous software tools. Developing procedures for efficiently interfacing their respective inputs and outputs in this project will serve as a foundation for future work on more realistic and efficient extensions to this numerical model for the atmospheric flow.

The software tools utilized in this project were selected based on their open-source availability, adaptability to our problem, and compatibility with each other for integration of the simulation pipeline.

The remainder of this paper will be organized as follows: Section 1 will outline the methods used for mesh generation, solving the compressible Euler equations, finite volume discretization, numerical models for the atmosphere simulation, and exporting the simulation data for visualization. In Section 2 we will present some simulation results and discuss their implications, including proposed directions for future extension of this work. In Section 3 we will make some concluding remarks.

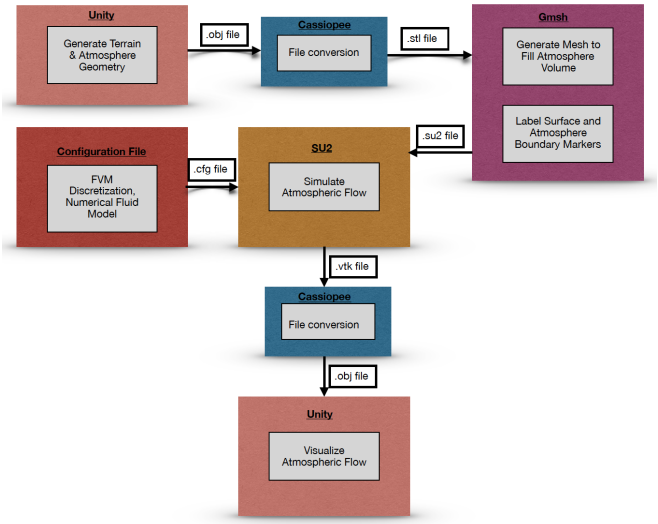

Fig. 1. Conceptual flowchart of the pipeline. Terrain is generated in Unity, then exported as a .obj file and converted to a .stl file by Cassiopee. Gmsh generates a tetrahedral mesh in the atmosphere volume and exports to a .su2 file which specifies the mesh geometry and boundary markers. This file is given as input to SU2 along with a configuration file specifying the simulation methods & parameters. After each time step of the simulation, a .vtk file is output and converted by Cassiopee back to a .obj file which can be visualized by Unity.

1. Methods

A. Software.

Three main software applications were used in order to perform the necessary tasks along our simulation pipeline. This process is illustrated in Fig. 1. Additionally, Cassiopee (CFD Advanced Set of Services In an Open Python EnvironmEnt) is used for interfacing in order to import terrain geometry from Unity to Gmsh and simulation output data from SU2 back into Unity. Cassiopee is a set of Python modules developed by Onera, providing advanced services for preparing CFD computations and post-processing CFD solutions (5). The use of this package for file conversion is discussed in more detail in F.R.'s report.

Unity is a cross-platform game engine for creating animations in two and three dimensions, as well as virtual reality. While best-known for its use in video games, being used in approximately half of new mobile games in 2018 (6), alternative applications including film, architecture, and automotive engineering (6).

Gmsh is an open-source 3D finite element mesh generation software. According to the developers, its goal is to provide a "fast, light and user-friendly meshing tool with parametric input and advanced visualization capabilities." (7)

Stanford University Unstructured (SU2) is an open-source collection of C++-based software tools for numerical analysis of PDEs, designed with computational fluid dynamics in mind. (8) One important feature in this software choice is the large community of scientists who use and adapt SU2 for their work. The ability to access and modify the source code allows greater flexibility in the development of our numerical models.

B. Mesh Generation.

Input planet terrain geometry is generated in Unity by adding noise to a sphere in order to create interesting topography. This is explored in more detail by F.R.'s report. Planet geometry is imported from Unity to Gmsh using Cassiopee. Details of this import process are also explored by F.R.'s report.

A larger, smooth sphere is created around this planet to form the outer boundary of our atmosphere as seen in Fig. 2. The resultant inner and outer shells are then labelled with boundary markers, allowing us to specify the planet surface and outer atmosphere and apply boundary conditions in the SU2 simulation.

Gmsh allows constructive solid geometry using Boolean operations which identify regions such as the intersection of two objects, their union, or their symmetric difference. By selecting the symmetric difference of the planet terrain shell and the larger spherical atmosphere shell we can obtain a domain that occupies only the annulus between the shells, forming our atmosphere.

This annular atmosphere domain is filled with a tetrahedral 3D mesh by Gmsh. The atmosphere mesh is now completed and ready to be used in a fluid simulation.

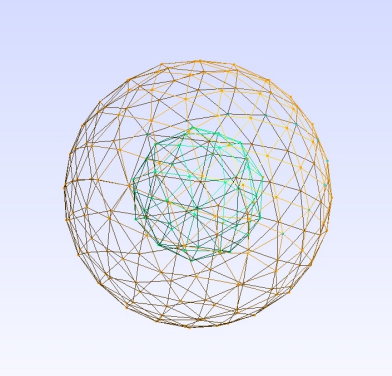

Fig. 2. Image of the basic spherical test terrain (green) with outer shell (orange). After Boolean selection of the annular region to form an atmosphere, 3D volume meshing will be performed.

The mesh is output as a .su2 file specifying the volume elements (tetrahedra), surface elements (triangles), and boundary elements labelled with the markers discussed above.

C. Compressible Euler Equations.

SU2 software can solve the compressible Euler equations, which can be expressed in differential form as:

$$\frac{\partial U}{\partial t} + \nabla \cdot \bar{F}^c(U) - S = 0 \qquad [1]$$

where the conservative variables are given by $U = (\rho, \rho\bar{v}, \rho E)^T$ and S is a generic source term. The convective flux $\bar{F}^c$ is defined as:

$$\bar{F}^c = \begin{Bmatrix} \rho\bar{v} \\ \rho\bar{v} \bigotimes \bar{v} + \mathbf{I}p \\ \rho E\bar{v} + p\bar{v} \end{Bmatrix}$$

where ρ is the fluid density, $\bar{v}$ is the vector flow speed, E is the total energy per unit mass, p is the static pressure, T is the temperature, and $\mathbf{I}$ is the identity tensor (8)

D. Discretization.

The Finite Volume Method is an approach to solving an integral conservation law by dividing the domain into control volumes, or cells, and approximating the conservation law on each volume.

As discussed in the Introduction, the Finite Volume Method approximates the integral form of a conservation law on each control volume in a meshed domain. Interpolation is used to express variable values at the control volume surface in terms of the centre values, and surface and volume integrals are approximated by numerical quadrature. (1)

In 1D, the integral form of a conservation law (with no source term) for a quantity U(x) in control volume i is:

$$\frac{d}{dt}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} U dx + F(U)|_{x_{i+\frac{1}{2}}} - F(U)|_{x_{i-\frac{1}{2}}} = 0 \qquad [2]$$

Defining the mean value of U in the control volume as:

$$U_i = \frac{1}{\Delta x_i}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} U dx \qquad [3]$$

where Δx_i is the length of the cell, and assuming that the solution in each volume is constant, we obtain a piece-wise continuous function:

$$\Delta x_i \frac{dU_i}{dt} + F(U)|_{x_{i+\frac{1}{2}}} - F(U)|_{x_{i-\frac{1}{2}}} = 0 \qquad [4]$$

In order to apply this method to a specific problem such as the compressible Euler equations, we calculate the flux of quantity U at each boundary of the cell ($x_{i+\frac{1}{2}}$ and $x_{i-\frac{1}{2}}$). We can then apply a time-discretization scheme to evaluate $U(x_i, t + \Delta t)$ in order to advance the solution in time. (9)

E. Numerical Model.

A number of central and upwind convective schemes are available in SU2 to calculate the convective fluxes. The difference between central and upwind schemes is easiest to explain in the one-dimensional case. A central scheme uses information from adjacent nodes on both sides of the point of interest when numerically approximating derivatives at a given node, whereas an upwind scheme uses only information from the nodes on one side depending on the direction of flow at that point. This direction is changed as necessary in order to make sure it captures information flowing towards the point of interest, hence the name "upwind". While upwind schemes can provide higher accuracy, we have chosen a central scheme as these are more stable and therefore allow larger time steps. (8)

The Jameson-Schmidt-Turkel (JST) scheme was chosen as a good compromise between accuracy and stability. The JST scheme is more accurate than the other central scheme available in SU2, the simpler Lax-Friedrich (LF) scheme which is very dissipative. (8) JST has second- and fourth-order scalar dissipation (10), while LF has a first-order dissipation term. (11)

The SU2 simulation is run with Euler wall conditions at each boundary (surface and outer atmosphere). This condition enforces zero normal velocity in the flow at the boundary, i.e. no fluid can flow into or out of the boundary wall. (12) While this is a good initial choice for approximating the boundary conditions at the planet surface, it is a less realistic approximation at the outer boundary. However, for a sufficiently deep atmosphere this model should roughly converge to the results of a model without an upper wall as the size of the atmosphere will be much larger than the size of the affected boundary layer.

Currently a second-order dual time-stepping method is used. Further investigation is needed into whether this is the best choice for balancing efficiency and accuracy, as an in-depth study of this method has yet to be conducted in the context of this project.

In order to model a planetary atmosphere, we will need to implement a gravitational source term with spherical coordinates. This has not yet been implemented, however relevant plans for developing this feature are discussed in Section **2.A**. Presently the default SU2 model of uniform downwards gravity in Cartesian coordinates is used, which is not physically correct for our problem but has been used as a placeholder in order to test the rest of the simulator features.

F. Simulation Data Output.

Output data from the SU2 simulation includes fluid density, speed, pressure, and temperature in the form of a .vtk file.

This data will be exported back into Unity for use in visualizing the atmospheric flows in virtual reality. (13)

The method for converting the .vtk file back to a .obj which can be input to Unity for visualization using Cassiopee is still under development, and falls primarily under the domain of F.R.'s project.

2. Results and Discussion

At the completion of this one-semester project, we have a method for producing planet surfaces with visually realistic terrain; parameters such as the planet size, atmosphere depth, mountain height and continent size can be controlled when generating the mesh. An example is displayed in Fig. 3.

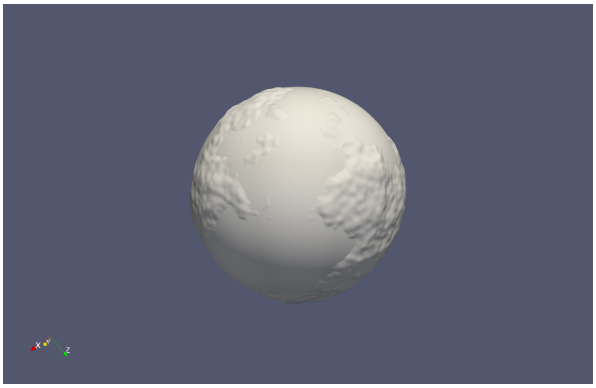

Fig. 3. Image of planet terrain generated in Unity, exported to Gmsh using Cassiopee. Topology such as mountains and valleys is introduced using noise to create a more realistic planetary model compared to the basic smooth sphere.

The integration of the terrain geometry into an SU2-readable 3D mesh is not yet completed; the file exported from Unity is currently defined as a 2D surface which must first be converted to a 3D volume before the outer shell can be added and the 3D mesh generated. As a placeholder for testing the rest of our method while we resolve this issue, a simplified geometry involving a smooth spherical planet terrain has been generated in Gmsh. This was done by producing two concentric spheres as seen in Fig. 2, then selecting the symmetric difference to select only the annulus between the shells as described in Section **1.B**.

Simulations using our placeholder planet terrain have been conducted using our Euler FVM solver as described in Section **1.E**. Fig. 4 displays two views of the results. Each of the images displays data for the momentum magnitude: a view at the outer edge of the atmosphere (upper) and a 2D slice through the middle of the planet.

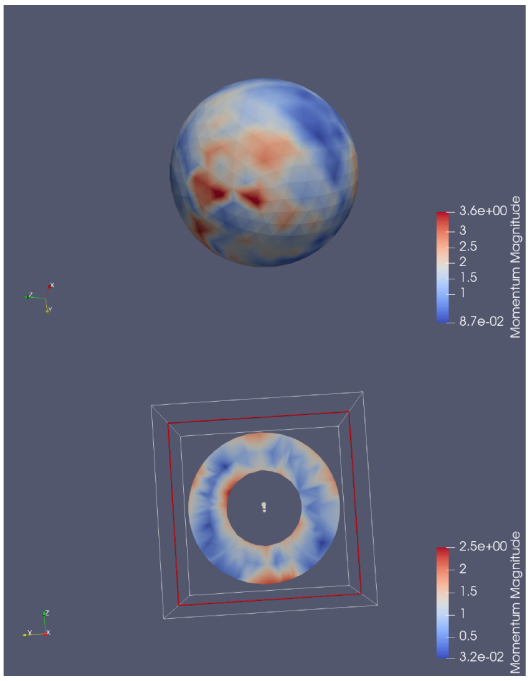

Fig. 4. Output data from preliminary test with smooth spherical terrain and initial numerical model. Color plot scales for each image is shown at bottom right, indicating the variable plotted and its visible range. **Top:** Momentum magnitude data plotted on the outer edge of the atmosphere. **Bottom:** Momentum magnitude data plotted in a 2D slice through the middle of the model.

These simulations can be run with relatively large time steps due to the large grid size and choice of stable numerical models as discussed in Section **1.E**; while time steps of order 0.001s are required for stability in many SU2 tutorials, this simulation remains stable even for steps as large as 1.0s. Stability for large time steps is one of the most important design requirements of this project since this will reduce the number of calculations required per second of live visual output.

Further investigation into what the exact limit for optimally balancing stability and accuracy as a function of time-step size should be is necessary to better understand the trade-offs involved once a final numerical model has been selected. However, as an initial proof-of-concept the results are promising and indicate that a real-time weather simulator using our approach is a realistic goal.

Once the simulation model has been further expanded to better capture the dynamics of a realistic atmospheric system, comparisons to real-world data as well as high-resolution simulations will be essential for validation purposes. These will demonstrate whether the approximations made in order to improve simulation speed produce results in acceptable agreement with the atmospheric flows they seek to model. A number of important directions for such expansion are discussed below.

A. Directions for Future Work.

A number of important features have not yet been fully developed at the end of our one-semester project, due in part to the extreme events of the Winter 2020 semester. In this section we present our plans and suggestions for the ongoing development of this real-time weather simulator. Further supporting materials in the form of N.L.'s lab notebook accompanied by a guide to the design and use of our framework have

also been compiled in order to aid future projects aiming to extend our work; this guide includes a list of resources related to proposed features.

In order to complete the integration of our simulation pipeline, we must address the problem presented earlier in Section 2 with respect to the conversion of our Unity terrain from a 2D surface to a 3D volume in Gmsh. Some examples of projects which have performed a similar conversion have been found in online CFD forums, indicating that this is not an insurmountable problem. Further information relating to these examples can be found in the supporting materials document.

The implementation of a gravitational source term with spherical coordinates is the most important step for the continued development of our numerical model. Some examples of the use of cylindrical coordinates in SU2 exist (14), and investigating these will likely enable modifications to produce our desired coordinate system. Additionally, it is possible to add a user-specified source term by modifying the C++ SU2 source code (8). This source term would point in the direction of the planet core and have a scalable strength which can be adjusted to model planets of varying size.

Once this more realistic source term and coordinate system are implemented, further work is needed into the specification of realistic initial conditions such as the 1D pressure and temperature profiles of the atmosphere as a function of altitude.

In order to better model the thermodynamics of the atmosphere, we are currently considering the use of boundary conditions at the outer atmosphere specifying a spatially uniform heat flux which varies as a function of time. This could take the form of a sinusoidal function in order to produce behaviour similar to the diurnal variation in solar flux seen by the Earth. While in reality the solar flux also varies as a function of latitude and longitude, this simple model should be a significant step towards realistic atmospheric behaviour.

Time-dependent heat flux should make a more significant improvement compared to our current wall boundaries in the outer boundary conditions than those at the surface. However, this may also be a useful model for the planet surface since solar radiation which penetrates the atmosphere is absorbed, causing time-dependent heating similar to the diurnal variation reaching the outer atmosphere. Further inquiry into the relative importance of each of these heat sources to atmospheric energy balances is needed to determine whether the proposed changes would be beneficial for our model.

In addition to the pre-packaged time-stepping methods provided in SU2 as discussed in Section **1.E**, other solvers could be investigated and implemented by modifying the SU2 source code. Alternative approaches used in atmospheric modelling literature include splitting the advection into a horizontal step and a vertical step since these directions usually have different characteristic velocities. This splitting allows each direction to be integrated with a time-step limited by their respective characteristic velocities, significantly improving computational economy by avoiding solving the original set of stiff equations. (15)

3. Conclusion

Initial investigations into the set-up of a real-time weather simulator using a simplified atmospheric flow model with SU2 software provide promising indications of feasibility. Visually realistic terrain can be generated in Unity and there are strong indications that this can be meshed in Gmsh. The resultant output mesh can be used to conduct fluid simulations in SU2 which will provide atmospheric flow data for visualization using Unity.

Solving the compressible Euler equations using the finite volume method with relatively large grid size produces a method which is stable even for the large time steps required to produce real-time simulations; this is one of the main goals of our project.

Avenues for future work include the implementation of a gravitational source term with spherical coordinates, more realistic boundary & initial conditions, and improvement of time-stepping methods. Validation of this improved model by comparison to real-world data and high-resolution simulations can then be performed to assess accuracy.

From a broader perspective, the global atmospheric flow simulator developed in this paper can be integrated into a larger project which includes solvers for modelling local weather microphysics and scattering & absorption in the atmosphere. Real geographical data could also be imported instead of our artificially-generated terrain, which could include data from exoplanets in addition to Earth.

The use of this weather simulator as a test case is intended to demonstrate the viability of applying game engine and virtual reality technology to the visualization of scientific data. This technology has the potential to improve the way in which data is visualized across a wide variety of domains, such as quantum chemistry, materials science, and nuclear physics.

ACKNOWLEDGMENTS. I would like to thank Dr. Hong Guo for guidance in the theory, software selection, and numerical model development throughout this investigation, and contribution to the contents of this paper. I would like to thank Filipe Rodrigues for his work in interfacing our two projects and assistance in using the various softwares required for this project. I would also like to thank Dr. Sabrina Leslie for guidance with regards to the presentation of the results in this paper.

4. References

1. J. Tu, G.-H. Yeoh, C. Liu (2018) Computational Fluid Dynamics: A Practical Approach, 3rd Edition *Butterworth-Heineman, Elsevier* pp. 1-29, pp. 163-168.
2. C. Yang, X.C. Cai (2013) A Fully Implicit Compressible Euler Solver for Atmospheric Flows. In: Bank R., Holst M., Widlund O., Xu J. (eds) *Domain Decomposition Methods in Science and Engineering XX. Lecture Notes in Computational Science and Engineering* Vol. 91. Springer, Berlin, Heidelberg
3. National Oceanic and Atmospheric Administration (2019) Global Circulations. *National Weather Service.*
4. C.D. Perez-Segarra et. al. (2006) Analysis of Different Numerical Schemes for the Resolution of Convection-Diffusion Equations using Finite-Volume Methods on Three-Dimensional Unstructured Grids. Part I: Discretization Schemes *Numerical Heat Transfer, Part B: Fundamentals* Vol. 49 Issue 4, pp. 333-350.
5. C. Benoit, S. Peron, S. Landier (2015) Cassiopee: a CFD pre- and post-processing tool. *Aerospace Science and Technology* Vol. 45, pp 272-283.
6. A. Oreskovic (2018) Why EA's former boss believes the 3D tech that powers video games will make way more money outside of gaming. *Business Insider.*
7. C. Geuzaine, J.-F. Remacle (2009) Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering* 79(11), pp. 1309-1331.
8. T.D. Economon (2016) SU2: An Open-Source Suite for Multiphysics Simulation and Design. *AIAA Journal* Vol. 54, No. 3.
9. K. Wilcox, Q. Wang (Spring 2014) Computational Methods in Aerospace Engineering. Massachusetts Institute of Technology: MIT OpenCouseWare, https://ocw.mit.edu/. License: Creative Commons BY-NC-SA.
10. R.C. Swanson, E. Turkel (1992) On Central Difference and Upwind Schemes. *Journal of Computational Physics* Vol. 101, pp. 292-306.
11. B. Seibold (Spring 2009) Numerical Methods for Partial Differential Equations. Massachusetts Institute of Technology: MIT OpenCouseWare, https://ocw.mit.edu/. License: Creative Commons BY-NC-SA.

12. N. Balakrishnan, G. Fernandez (1998) Wall boundary conditions for inviscid compressible flows on unstructured meshes. *International Journal for Numerical Methods in Fluids* Vol 28, pp. 1481-1501.
13. M. Berger, V. Cristie (2015) CFD post-processing in Unity3D. *Procedia Computer Science*. Vol 51., pp. 2913–2922.
14. T.D. Economon (2018) Simulation and Adjoint-based Design for Variable Density Incompressible Flows with Heat Transfer. *American Institute of Aeronautics and Astronautics, Multidisciplinary Analysis and Optimization Conference*.
15. A.J. Gadd (1978) A split explicit integration scheme for numerical weather prediction. *Q.J.R. Meteorol. Soc*. Vol. 104, pp. 569-582.