# *Remove Before Takeoff*: A Configuration Aerodynamics Video Game

Amaru A. Ordóñez-Jacobson*
*CS399: Independent Study, Department of Computer Science, Stanford University, Stanford, CA*

## I. Introduction

THERE exist very few easily accessible ways for people to learn about aeronautics. Despite airline ridership steadily increasing year over year, public interest in topics relating to aeronautics remains low despite local and national governmental initiatives to encourage engagement and careers in aeronautics [1] [2]. NASA's own *Beginner's Guide to Aeronautics* looks like it was designed in the early 2000's because it literally was, and to this day remains one of the only publicly accessible resources for learning about how planes fly (Fig. 1).

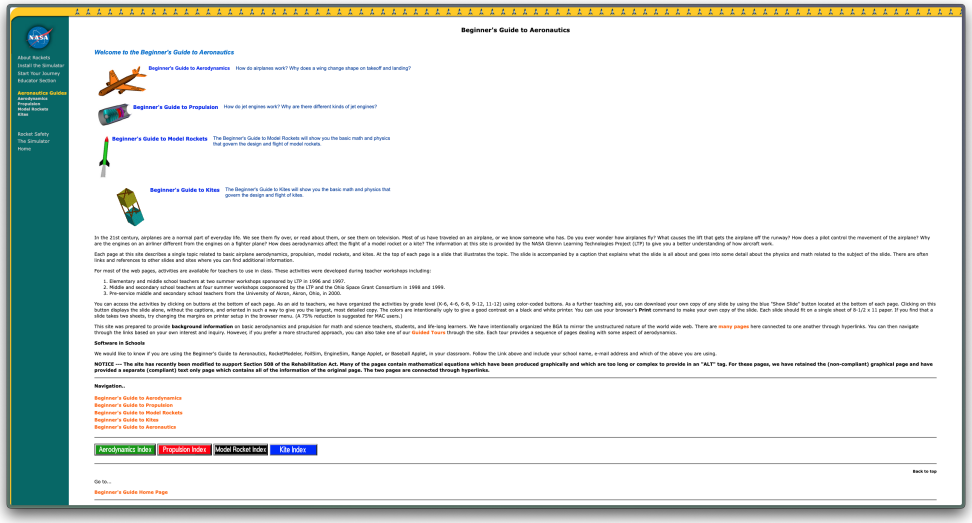


**Fig. 1 A screenshot of the landing page for NASA's *Beginner's Guide to Aeronautics*. A fantastic tool for its time, but incredibly difficult to find and no longer as accessible [3].**

This difficulty of access and ease of use for avenues to learn about aeronautics contributes to decreased interest. In my own experience both as an undergraduate and graduate student in the Department of Aeronautics and Astronautics at Stanford University, I have often found myself and my peers asking, "How does anyone ever fully understand what's going on here?" and "Why do I always feel like I'm catching up?"

During the Autumn of 2025, I took an advanced game design class at Stanford (CS377G). For the first project in this course, our groups were tasked with designing a learning game that would teach players how to complete a certain task. Using the philosophy from this project, as well as the aforementioned concerns regarding contemporary aeronautics education, I proposed the present work – an educational video game targeted at students with a fundamental understanding of physics that encourages players to build an intuitive understanding of configuration aerodynamics.

This report details the design decisions for this game, *Remove Before Takeoff* (*RBT*), the methodology behind its physics simulator, and its playtest and design history. Although this game is not complete, this report will also function as a stepping stool as I continue implementing more features into the game.

*M.S. Candidate, Aeronautics and Astronautics, Department of Aeronautics and Astronautics, Stanford University. amaru@stanford.edu

## II. Preliminary Design Decisions

Starting this project was difficult as I was largely on my own with very little experience building physics simulators in 3D environments. As with any project, the way we make progress is by starting small and tackling one problem at a time. Scouring the internet for inspiration, I eventually found a game that accomplished a similar task to the one I set for myself called *Kerbal Space Program* (*KSP*). *KSP* is similar to my proposed game in a lot of ways – it encourages people to get excited about interesting engineering problems in a way that is fun and silly enough to garner an immense audience of laypeople but rigorous enough to receive attention and commendation from large national space agencies such as NASA [4].

For the first two weeks of this project, I played *KSP* nearly every day to understand firsthand its mechanics, aestheics, its types of fun, and the elements of the systems it represents it includes and excludes. With regards to this last point, after some searching online, I found a copy of the physics documentation of *KSP* [5]. After reading through it and discussing with my advisor, I settled on one of the guiding philosophies behind this project:

> "A little unrealism creates forgiveness."

Using this mantra in combination with the physics documentation and experience of playing *KSP*, I then set out to build a moodboard to visualize and concretize my design intentions (Fig. 2).

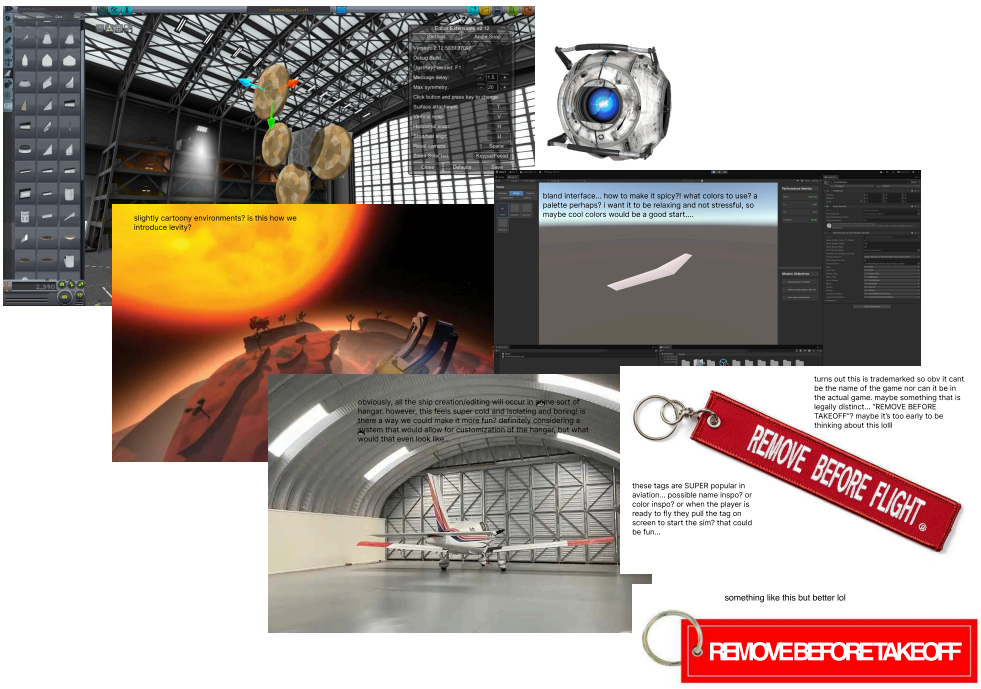


**Fig. 2 A first iteration of the moodboard used to guide this project.**

Finally, I formalized the criteria that I would use to create the game engine and gameplay within three groups:

- **Physics.**
  The proposed game will represent more granular physics than *KSP*. The reason for this is twofold – not only is our focus purely on atmospheric flight (as opposed to atmospheric and space), but we also have access to more advanced processors that can perform computations faster and more precisely. An example of this increased granularity can be found in *KSP*'s documentation on the drag force calculation, which is simply "directly proportional to the cross-sectional area A of the ship" [5]. Although this is a great place to start, there are other elements of drag (e.g. vortex drag) that are completely ignored under these kinds of assumptions.
- **Aesthetics.**
  *KSP* is very cutesy, silly, and full of the aforementioned unrealism that begets forgiveness. However, this game has a different focus and therefore a distinct aesthetic. The focus in this game is on the *learning goals* of the game. *KSP* is mostly about exploring the different and quirky ways one can design rockets and spaceplanes,

allowing them to perform strange and fun missions. This game is focused on giving players practical phyiscal intuition of configuration aerodynamics, and a split focus on silly characters and serious physics might leave both unsatisfactory.

- **Goals and Design Customization.**
  As I played through *KSP*, I realized that, although there is a wide variety of mission types, many mission requirements (especially in the early game) are something like "Contain X booster stages, winglets, and a parachute." A mission in this game would include more specific requirements regarding aerodynmaic performance parameters such as the lift coefficient, specific ranges, and cruise velocities. Additionally, *KSP*'s customization options, while extensive, don't allow users to edit the *geometry* of parts in a design. This is one of the primary goals of this project – to allow players to edit part geometry and see its effect on the performance of the airplane in real time.

With all of this in mind, I finally set out to create MVPs of the two main interfaces of this game – the ship editor and the simulator. This quarter, I was also able to implement a tutorial mission (although there are still sections that require work).

## III. Physics Simulation and Performance Estimation

This section will serve as a bit of physics documentation and contain notes for myself as I continue this project into next quarter. The vast majority of the physical relationships implemented are based on Richard Shevell's *Fundamentals of Flight* [6]. The simulator uses a 2D point-mass equations-of-motion model augmented with quasi-3D aerodynamics (finite-wing lifting-line theory and Shevell component buildup drag). Angular dynamics are modeled separately for pitch and roll via spring–damper systems.

### A. Atmosphere (ISA)

Air density, temperature, and viscosity vary with altitude $h$ (metres) using the International Standard Atmosphere troposphere model.

*1. Density*

$$\rho(h) = \rho_0 \left(1 - 2.26 \times 10^{-5}\, h\right)^{4.256} \qquad \rho_0 = 1.225 \text{ kg/m}^3 \tag{1}$$

*2. Temperature*

$$T(h) = T_{\text{ref}} \left(1 - 2.2558 \times 10^{-5}\, h\right) \qquad T_{\text{ref}} = 288.15 \text{ K} \tag{2}$$

*3. Speed of sound*

$$a(h) = \sqrt{\gamma R\, T(h)} \qquad \gamma = 1.4, \quad R = 287.05 \text{ J kg}^{-1}\text{K}^{-1} \tag{3}$$

*4. Dynamic viscosity (Sutherland's law)*

$$\mu(T) = \mu_{\text{ref}} \left(\frac{T}{T_{\text{ref}}}\right)^{3/2} \frac{T_{\text{ref}} + C}{T + C} \qquad \mu_{\text{ref}} = 1.789 \times 10^{-5} \text{ Pa s}, \quad C = 110.4 \text{ K} \tag{4}$$

**Assumption:** Only the ISA troposphere lapse rate is modeled; the stratosphere (above $\approx 11$ km) is not treated separately.

### B. Aerodynamic Forces

*1. Dynamic pressure*

$$q = \tfrac{1}{2}\, \rho\, V^2 \tag{5}$$

*2. Angle of attack*

The angle of attack is the difference between pitch angle $\theta$ (including trim $\delta$) and the flight-path angle $\gamma$:

$$\alpha = (\theta + \delta) - \gamma \tag{6}$$

*3. Lift coefficient (Prandtl finite-wing lifting-line)*

The 2D thin-airfoil lift slope $a_0 = 2\pi$ rad$^{-1}$ is corrected for finite span using the Prandtl lifting-line result:

$$a = \frac{a_0}{1 + \dfrac{a_0}{\pi \cdot \mathrm{AR}}} \qquad a_0 = 2\pi \tag{7}$$

The lift coefficient including zero-lift angle $\alpha_0$ (positive for cambered wings) is:

$$C_L = a\,(\alpha - \alpha_0) \tag{8}$$

*4. Stall model*

Stall onset occurs at $|\alpha| > 15°$. Over a 10° transition band the $C_L$ limit is interpolated linearly from $C_{L,\max} = 1.8$ down to $0.4\,C_{L,\max}$:

$$C_{L,\mathrm{lim}} = \mathrm{lerp}\!\left(C_{L,\max},\ 0.4\,C_{L,\max},\ \frac{|\alpha| - 15°}{10°}\right) \qquad |\alpha| \in [15°, 25°] \tag{9}$$

*5. Parasite drag coefficient*

For each aircraft component $i$:

$$C_{D_p} = \sum_i K_i\, C_{f,i}\, \frac{S_{\mathrm{wet,i}}}{S_{\mathrm{ref}}} \tag{10}$$

*6. Skin friction (Schlichting turbulent flat plate)*

$$C_f = \frac{0.455}{\left(\log_{10} Re\right)^{2.58}} \qquad Re = \frac{\rho V \ell}{\mu} \tag{11}$$

where $\ell$ is the mean chord (lifting surfaces) or body length (fuselages).

*7. Wing/tail form factor*

$$K_{\mathrm{wing}} = 1 + Z\,\frac{t}{c} + 100\left(\frac{t}{c}\right)^4 \qquad Z = \frac{(2 - M^2)\cos\Lambda}{\sqrt{1 - M^2\cos^2\Lambda}} \tag{12}$$

where $t/c$ is the thickness ratio and $\Lambda$ is the quarter-chord sweep angle.

*8. Fuselage form factor (Shevell curve-fit at $M = 0.5$):*

$$K_{\mathrm{body}} = 1 + \frac{0.57}{f} + \frac{3.80}{f^2} \qquad f = \frac{L}{D}\ \text{(fineness ratio)} \tag{13}$$

*9. Induced drag coefficient*

$$C_{D_i} = \frac{C_L^2}{\pi\,\mathrm{AR}\,e} \tag{14}$$

The Oswald span-efficiency factor $e$ is derived from the Shevell formula:

$$e = \frac{1}{\pi\,\mathrm{AR}\,k\,C_{D_p} + \dfrac{1}{u\,s}} \qquad u = 0.99, \quad s = 0.975 \tag{15}$$

where $k$ is an area-weighted sweep factor (0.38 unswept; 0.45 at 35 sweep).

*10. Lift and drag forces*

$$L = q\,C_L\,S \qquad D = q\,(C_{D_p} + C_{D_i})\,S \tag{16}$$

### C. Equations of Motion

*1. Ground roll (takeoff)*

Before lift-off ($L < W$, $h = 0$):

$$\frac{dV}{dt} = \frac{T - D - \mu_r(W - L)}{m} \qquad \mu_r = 0.02 \text{ (dry concrete)} \tag{17}$$

*2. Airborne point-mass dynamics*

Once airborne, the aircraft is treated as a point mass. Banking by roll angle $\phi$ tilts the lift vector, so only its vertical component $L\cos\phi$ opposes gravity:

$$\frac{dV}{dt} = \frac{T - D}{m} - g\sin\gamma \tag{18}$$

$$\frac{d\gamma}{dt} = \frac{L\cos\phi - W\cos\gamma}{mV} \tag{19}$$

$$\frac{dh}{dt} = V\sin\gamma \tag{20}$$

$$\frac{dx}{dt} = V\cos\gamma \tag{21}$$

**Assumption:** The lateral ($y$) degree of freedom is not tracked; rolling induces a turn rate consistent with the banked-lift decomposition but the lateral position is ignored. This keeps the simulation effectively 2D (altitude + forward distance).

### D. Angular Dynamics

*1. Pitch*

Pitch angle $\theta$ is driven by a spring toward the autopilot target $\theta^*$, disturbed by the aerodynamic moment from the centre-of-mass fore-aft offset $\Delta z_{\text{CoM}}$, and damped:

$$\ddot{\theta} = \left[K_p(\theta^* - \theta) - K_s\,\Delta z_{\text{CoM}}\,L - K_d\,\dot{\theta}\right]\frac{I_{\text{ref}}}{I_{\text{pitch}}} \tag{22}$$

where $I_{\text{ref}} = 5000\ \text{kg m}^2$ is a normalisation inertia and $I_{\text{pitch}}$ is the estimated pitch moment of inertia. A positive $\Delta z_{\text{CoM}}$ (CoM ahead of aerodynamic centre) produces a nose-down restoring moment (stable configuration).

*2. Roll*

Roll is driven by the lateral CoM gravity torque and a dihedral restoring torque:

$$\tau_{\text{grav}} = -\frac{\Delta x_{\text{CoM}}\,m\,g}{I_{\text{roll}}} \tag{23}$$

$$\tau_{\text{spring}} = -\frac{\phi\,k_{\text{dih}}\,L}{I_{\text{roll}}} \tag{24}$$

Damping is applied via exact implicit decay (unconditionally stable):

$$\dot{\phi}_{n+1} = (\dot{\phi}_n + (\tau_{\text{grav}} + \tau_{\text{spring}})\,\Delta t)\cdot e^{-k_{\text{damp}}\,\Delta t/I_{\text{roll}}} \tag{25}$$

The dihedral restoring coefficient $k_{\text{dih}}$ is computed by `SimulatorManager` as an area-weighted sum over wing panels:

$$k_{\text{dih}} = \frac{\sum_i b_i \sin \Gamma_i \, S_i}{4 \, S_{\text{total}}} \tag{26}$$

where $b_i$, $\Gamma_i$, $S_i$ are the span, dihedral angle, and area of panel $i$.

### E. Summary of Key Assumptions and Equations

- ISA troposphere only; no stratosphere model.
- Fully turbulent boundary layer (Schlichting $C_f$); no laminar correction.
- Thin-airfoil $a_0 = 2\pi$; no Mach compressibility correction to lift slope.
- Lifting-line theory (elliptical span-load); non-elliptical corrections absorbed into the Oswald efficiency $e$.
- Point-mass EOM; no pitching-moment or yawing-moment equations of motion (replaced by the decoupled spring–damper pitch/roll model).
- Lateral position is not integrated; the aircraft flies a straight track in the forward–vertical plane regardless of roll angle.
- Stall is modeled as a smooth $C_L$ roll-off; no post-stall drag spike or pitching-moment reversal.
- Thrust is assumed to act along the flight path (no incidence angle between thrust axis and velocity vector).

## IV. Playtesting and Version History

Due to the amount of time this quarter spent on building the physics engine and design editor, there unfortunately was very little time for formal playtesting. Before discussing playtesting, I would first like to list out the milestones achieved in development this quarter.

- **28 January, 2026**
  The MVP of the editor was completed. Although it was very barebones, here I was able to implement:
  1) Drawing the UI elements.
  2) Populating the UI with a grid of prefab parts for testing.
  3) Hovering over elements in the menu bring up a popup that gives a short summary of the element.
  4) Clicking on elements would "pin" the popup to the screen.
- **17 February, 2026**
  The editor was cleaned up (Fig. 3) and I added more performance metric functionality:
  1) Menus appear in their correct positions 100% of the time (yes, this was an issue).
  2) Performance metrics populate and update live as parts are placed in the viewport.
  3) Element rotation and translation fully implemented.
  4) WASD camera controls and scroll zoom implemented.
- **19 February 2026**
  Once the UI of the editor was ready, it was time to actually implement active player design. First, we start with wing design. Although I wanted to use prefab wing assets found online, after some searching I realized that these assets really do not exist. As such I decided to implement a handful of useful functions that would just model the wing for the player after being given some inputs. This is possible because, after all, an airfoil is parametrized by simple functions. We start with the NACA-series airfoils first, as they are the easiest to understand. By this point we've implemented:
  1) A full custom wing designer (Fig. 4). A player inputs a NACA number that is used for the airfoil.
  2) A wing editor that uses sliders to allow players to change geometric parameters such as wing span, taper, sweep, etc.
  3) A live performance metric update on the wing editor that shows the player how the performance of the wing changes in real time.
  4) After inputting all fields, the game creates an asset and populates it in the viewport, also updating the live total design performance metrics.
  5) A full custom fuselage designer inspired by the wing designer. Not as high-fidelity, but still usable.
- **24 February to 11 March 2026**
  From here, all updates were made to the simulator. Little by little I added assumptions, equations, and quality of life changes to make the game feel more like *a game*. Additionally, it was during this time period that the current name of the game was settled on – *Remove Before Takeoff*. It's not completely decided on, but this name

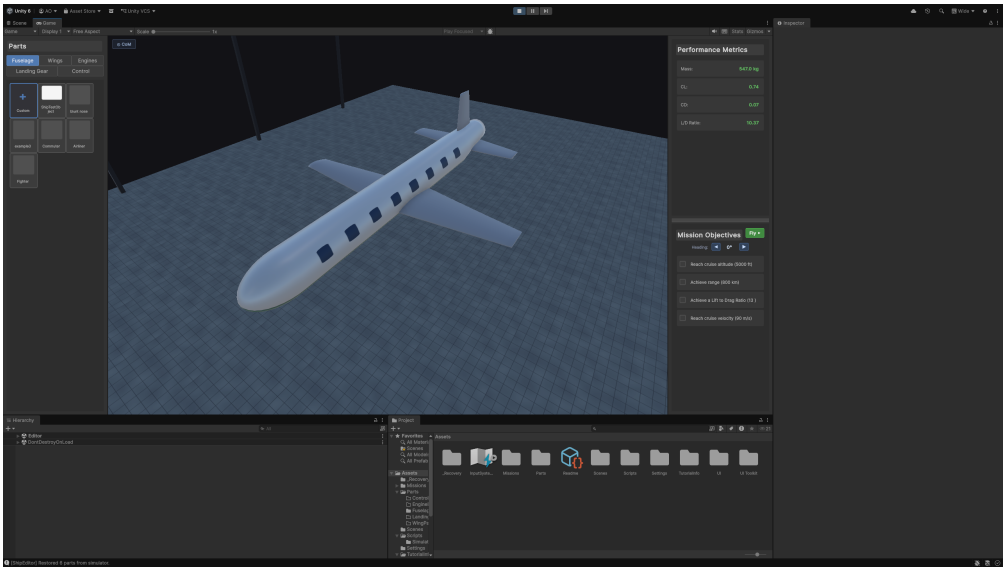


**Fig. 3 The current version of the editor in Unity.**

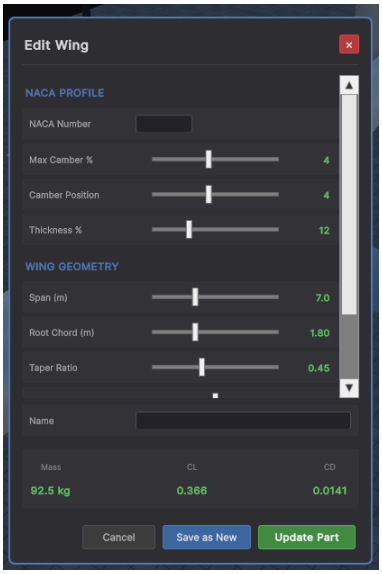


**Fig. 4 Full custom wing editor implementation with sliders and performance metrics at the bottom.**

is interesting as it references one of the most iconic and recognizable icons of aeronautics, the red *REMOVE BEFORE FLIGHT* tags. These tags are used on everything from airplane doors to landing gear, and I found it interesting that in a game about designing airplanes, it would be interesting to not only have a tactile tag removal to start the simulation (Fig. 5), but also serve as a slightly cheeky reference to the corrections the player needs to make to existing designs for the first few missions of the game.

- **11 March 2026**
  The tutorial mission was implemented, along with the light narrative structure to introduce unrealism into the game. The narrative is very barebones for now, but the premise is that the player is a QA intern that is fixing bad designs left by a previous engineer. Although at first it seems like hte engineer was just bad at their job, I want to imply that their poor performance was active sabotage for reasons I have yet to come up with. You are introduced to the game by a robot named T.B.D. (whose final name is to be determined), who walks you through the steps of interacting with the editor, changing airplane geometry, and running the simulator.

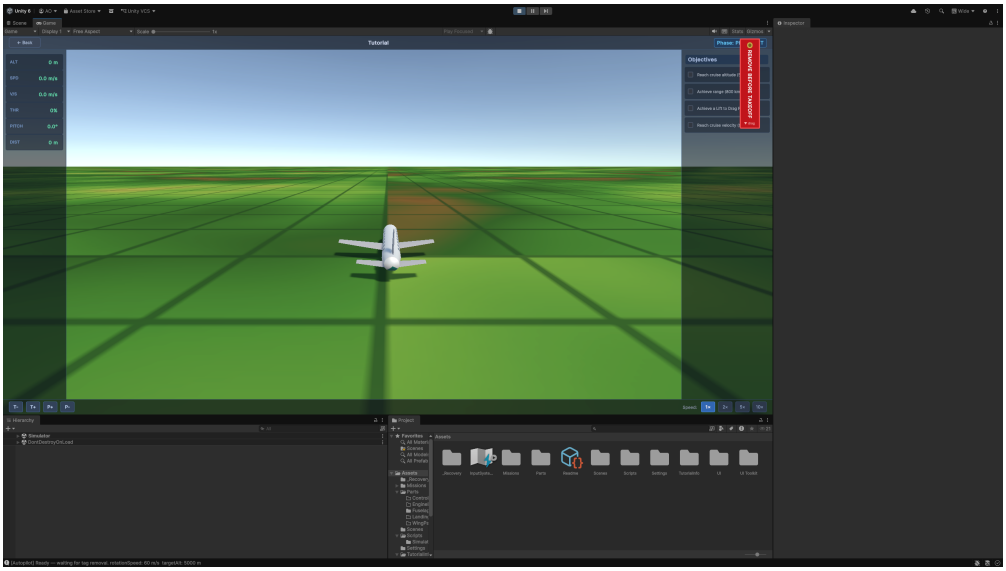

**Fig. 5 The current version of the simulator in Unity. Note the *REMOVE BEFORE TAKEOFF* tag in the top right that must be clicked and dragged down to start the simulation.**

Finally, I would like to describe the feedback I've received so far in playtesting that I hope to incorporate into the next version of the game. I was mostly interested in testing the editor UI, but also wanted feedback on the pacing of onboarding and the engagement of the player. Although my playtesters have yet to be within my aforementioned target demographic (people with a baseline knowledge of physics and possibly some familiarity with aeronautics), I was still able to get fantastic feedback regarding the UI. Some suggestions were:

1) Increase text size. Overall, text size is a little small and it's difficult to read everything on screen. There is a lot of information to take in while designing an airplane, and it would be helpful if we didn't need to crane our necks to read every piece of text.
2) Add visualizations to the wing and fuselage editor. Although performance metrics update live, it's difficult to really visualize what the design will look like after its changes, especially if the changes are significant. An incredible suggestion was made to render a side and top view of the wing, for instance, while moving the sliders in order to better visualize the geometric changes.
3) Allow direct user input in the wing editor. In its current state, the wing editor mostly allows input exclusively through sliders, with the only true typed user input in the NACA number input field. Allowing users to directly input their desired span, twist, etc. would greatly increase usability of the wing editor.

These pieces of feedback were incredibly helpful and have encouraged changes that are already in progress for the next version of the game.

## V. Conclusion

In an effort to address a lack of engaging resources for people to learn more about and have fun with the principles of aeronautics, *Remove Before Takeoff* is a learning game that aims to provide a fun and challenging environment for players to not only learn to love airplane design but also build an intuitive understanding of airplane geometry. This quarter, the foundations of the editor, simulator, and narration were implemented, as well as a good portion of the physics engine that will run during the simulation in real time. Although there are issues with the physics engine (for instance, moments seem to be computed incorrectly and planes often refuse to roll when obviously unstable), the fundamental math behind the engine is correct and will slowly become more realistic. However, this game is not without its own unrealism, as we build a narrative that introduces levity and perhaps a bit of comedy in order to beget forgiveness from the player.

For next steps, I first plan on finalizing the physics engine, implementing changes to ensure moments are calculated correctly across the airplane. I will then focus on mission design and exact learning goals, and will receive advice from professors in the Stanford Aero/Astro Department in order to determine these goals. Finally, I would like to improve the

visual fidelity of the game, particularly in the simulator environment, in order to introduce just a little more unrealism.

## Acknowledgments

I would like to thank my advisor, Prof. Christina Wodtke, for encouraging me to pursue this independent study in the first place and for keeping me going when it became difficult to see progress in this project. I would also like to thank my peers in the Winter 2026 cohort of CS399 – my last year at Stanford wouldn't have been the same without watching all of your projects grow and change throughout the quarter.

# References


[1] Bureau of Transportation Statistics, ., "U.S. Air Carrier Aircraft Departures, Enplaned Revenue Passengers, and Enplaned Revenue Tons," , 2019. URL https://www.bts.gov/content/us-air-carrier-aircraft-departures-enplaned-revenue-passengers-and-enplaned-revenue-tons.

[2] California Department of Transportation Division of Aeronautics, ., "California Public Utilities Code Section 21001 et seq. relating to the State Aeronautics Act," , Mar. 2019. URL https://dot.ca.gov/-/media/dot-media/programs/aeronautics/documents/puc_ssa_r3_2019.pdf.

[3] Benson, T., "Beginner's Guide to Aeronautics," , 2000. URL https://www.grc.nasa.gov/www/k-12/VirtualAero/BottleRocket/airplane/.

[4] Hall, C., "To the Mun and back: Kerbal Space Program," , Jan. 2014. URL https://www.polygon.com/features/2014/1/27/5338438/kerbal-space-program/.

[5] Kerbin, M., "Documentation of KSP Physics Version 1.2.1," , Oct. 2014. URL https://github.com/mhoram-kerbin/ksp-physics-documentation.

[6] Shevell, R. S., *Fundamentals of flight*, Prentice-Hall, 1989.