Teaching modeling using simulation and Tello drones

The tello_sim Python package is a simulator designed for use in classrooms using the Tello flight API

The package supports the Next Generation Science Standards related to “Developing and Using Models”.

Middle School Standards

Modeling in 6–8 builds on K–5 experiences and progresses to developing, using, and revising models to describe, test, and predict more abstract phenomena and design systems.

- Evaluate limitations of a model for a proposed object or tool.
- Develop or modify a model—based on evidence – to match what happens if a variable or component of a system is changed.
- Use and/or develop a model of simple systems with uncertain and less predictable factors.
- Develop and/or revise a model to show the relationships among variables, including those that are not observable but predict observable phenomena.
- Develop and/or use a model to predict and/or describe phenomena.
- Develop a model to describe unobservable mechanisms.
- Develop and/or use a model to generate data to test ideas about phenomena in natural or designed systems, including those representing inputs and outputs, and those at unobservable scales.

https://ngss.nsta.org/Practices.aspx?id=2

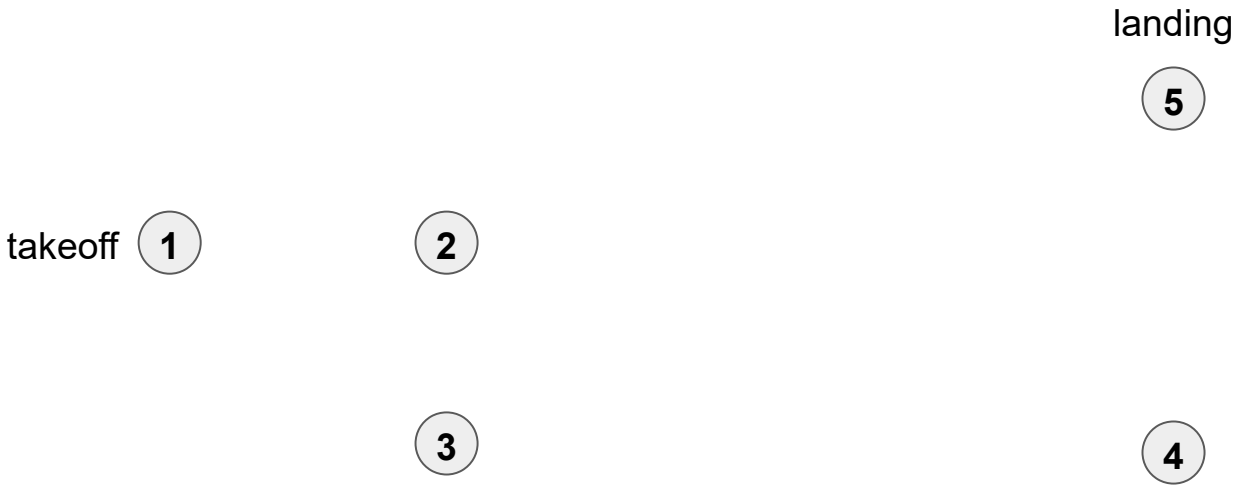
High School Standards

Modeling in 9–12 builds on K–8 experiences and progresses to using, synthesizing, and developing models to predict and show relationships among variables between systems and their components in the natural and designed world(s). Evaluate limitations of a model for a proposed object or tool.

- Evaluate merits and limitations of two different models of the same proposed tool, process, mechanism, or system in order to select or revise a model that best fits the evidence or design criteria.
- Design a test of a model to ascertain its reliability.
- Develop, revise, and/or use a model based on evidence to illustrate and/or predict the relationships between systems or between components of a system.
- Develop and/or use multiple types of models to provide mechanistic accounts and/or predict phenomena, and move flexibly between model types based on merits and limitations.
- Use a model to provide mechanistic accounts of phenomena.
- Develop a complex model that allows for manipulation and testing of a proposed process or system.
- Develop and/or use a model (including mathematical and computational) to generate data to support explanations, predict phenomena, analyze systems, and/or solve problems.

https://ngss.nsta.org/Practices.aspx?id=2

- A typical project can start with a drone course that is laid out in the classroom. Students then work together to develop a flight plan based on measuring the waypoints in the course.
- Obstacles can be added to make the course more challenging.
- The class and be divided into teams depending on the time you have and your instructional objectives.



Using Python in a Jupyter Notebook

- Jupyter Notebooks are a widely used tool for teaching programming. There are many resources including a dedicated book about using Jupyter Notebooks for education- https://jupyter4edu.github.io/jupyter-edu-book/index.html
- The easiest way to use the tello_sim package in a classroom is to use the Launch Binder link on the tello_sim Github page- https://github.com/Fireline-Science/tello_sim
- Note: Binder works for running the simulation, but can not be used to deploy to a drone. Jupyter Notebook and the Python packages must be locally installed to connect to the drone.

Once you have a launched Jupyter Notebook, students use their measurements of the course to create a programmed flight path. The flight path is entered into the simulation software using the Tello flight commands. There are many different options for a flight plan. In the program below, the altitude is not adjusted and the drone does not rotate.

```
import tello_sim
```
Python

```
my_drone = tello_sim.Simulator()
```
Python

```
my_drone.takeoff()
```
Python

```
my_drone.move_forward(100)
```
Python

```
my_drone.flip("f")
```
Python

```
my_drone.rotate_counter_clockwise(180)
```
Python

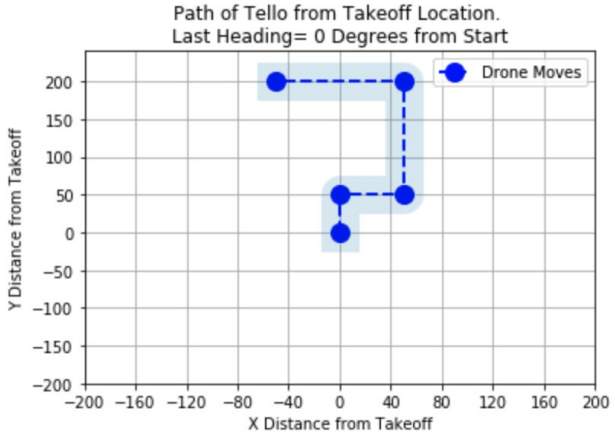
```
my_drone.move_forward(40)
```
Python

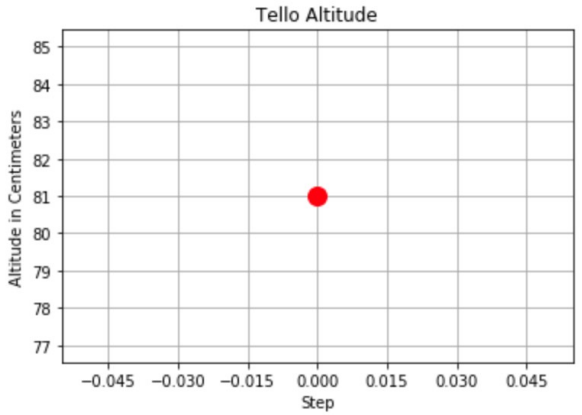
```
my_drone.move_up(25)
```
Python

When the simulation is run, the program plots out each step of the program. The simulator also insures that the program syntax is correct and that flight rules are followed (e.g.- the “takeoff” command must be executed before any flight commands).

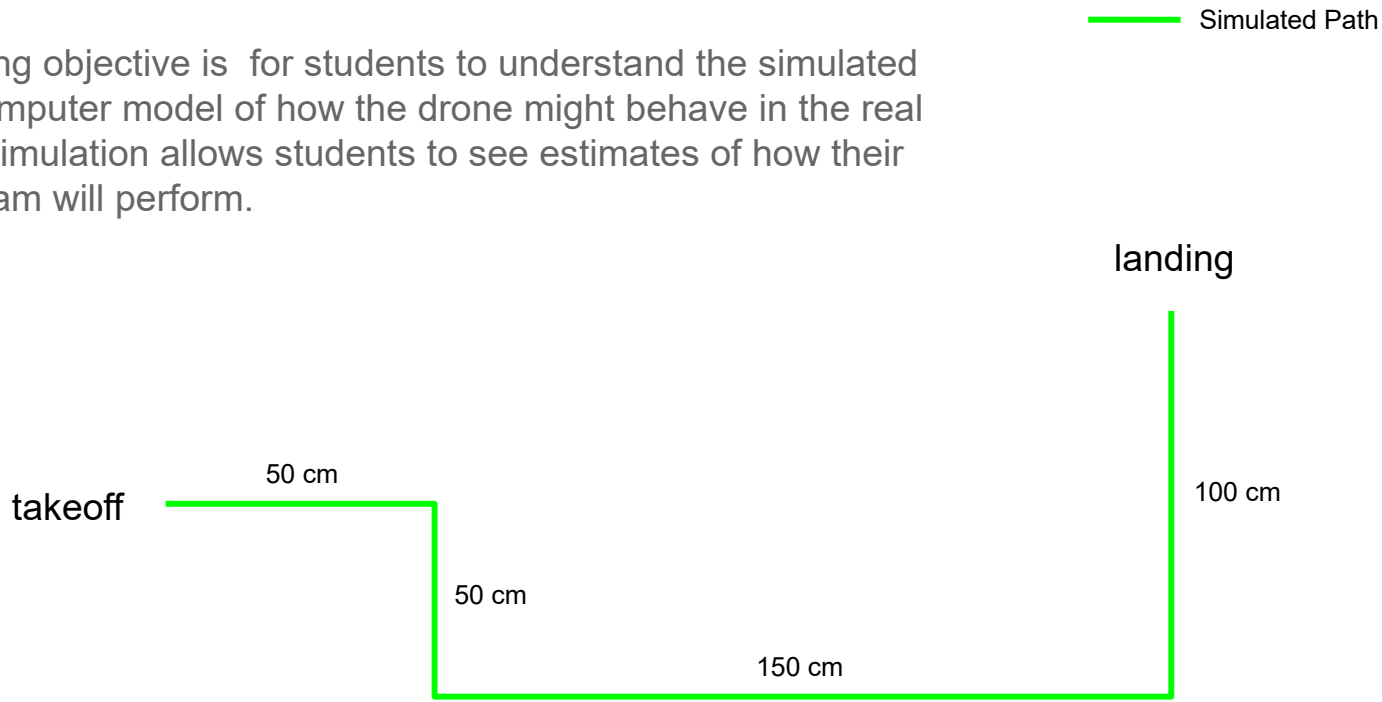
The standard simulation includes a light blue error bar of 25 centimeters.

```
Get ready for landing!
I am flying at 81 centimeters above my takeoff altitude.
I am running your "land" command.
Here are the graphs of your flight! I can't wait to try this for real.
```





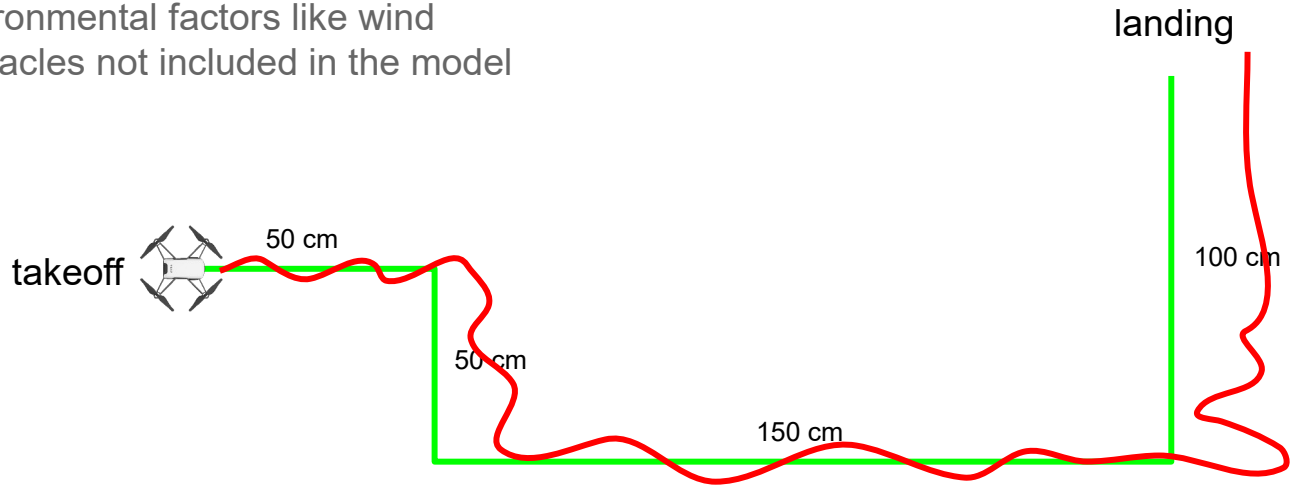
A key learning objective is for students to understand the simulated flight is a computer model of how the drone might behave in the real world. The simulation allows students to see estimates of how their drone program will perform.



The model is tested when the program is deployed to the drone.

The actual flight will differ from the simulated flight depending on a variety of factors including:

- The programmed flight path
- Accuracy and precision of drone flight computer tracking
- Environmental factors like wind
- Obstacles not included in the model



Deploying Programs to the Drone

- If you want to control when the programs are deployed to the drone, you can require students to share their saved program using the `drone.save(file_path='save_file.json')` option in the Jupyter Notebook.
- You can then load each students program using the `drone.load_commands(file_path='new_commands.json')` option. Details of this are available at https://github.com/Fireline-Science/tello_sim
- You can also let students deploy their own programs as long as they are working with a computer that can connect to the drone WiFi.
- If you have multiple drones, it is useful to write their WiFi network name on the drone to keep track of where you are connecting.

Each drone has a unique WiFi name that starts with “tello” and ends with a unique identifier.

After testing the simulated flight by flying the drone, students can discuss:

- If the drone was flown multiple times, were there differences between the flights?
- Why might there be differences between flights if the program remains the same?
- How accurate was the simulated flight compared to the real one?
- Where did the actual flight differ from the simulated flight?
- What are some reasons for these differences?
- How could the simulation be made more accurate?
- How could the flight program be adjusted to better match the planned flight path?

The programmed flight plan and actual flight can be updated to include:

- Adjustments to the flight variables based on the actual flight (e.g.- adjusting flight distances)
- Changes to the flight path to account for obstacles (e.g.- fly higher to avoid a chair)
- Changes to the environment (e.g.- where is the drone located and how it is oriented, shielding the drone from the wind, etc.)
- Changes to the error bar for plotting