**Notes on designing for the motion simulator (Voyager Dual-Seat Edition)**

* **The Voyager can be anything you imagine.**It may look like a truck from the outside, but inside it is an immersive black-box theater. Anything that moves can be simulated (more or less).
* **Players have limited time.**The runtime clock is controlled by the motion system and is currently set at 4.5 minutes. This is as long as an average person might remain comfortable. Some will last less, but developers have sat in the system for hours at a time when testing builds, so your mileage may vary. This is also a balance of what the industry calls "throughput," how many people can go through the attraction in an hour. Longer ride cycles reduce throughput which reduces revenue.
* **Players have to 'get it' right away.**Because of the limited time in the attraction, the players need to understand their roles and controls within the first 5-10 seconds. The experience can ramp up from there, but they have to be engaged and empowered immediately.
* **All players need to be engaged.**This is a two-seater platform. Experiences should generally be two-player cooperative - remember both seats experience the same motion profile so they are literally in the same boat together (or car, plane, body, or whatever your story puts them inside).
* **Don't forget the people in the queue.**While waiting in line to ride, they can be learning about the experience - learning the controls, story, or other info that will help them get into the story and their roles faster. They can also play along or against the people inside the motion platform using external computers or mobile devices. Extend the experience and increase the audience engagement.
* **Get creative with controls.**The system had a steering wheel and a throttle with one thumb button by default. Wireless XBox360 game controllers are available for both seats. Alternate controls are possible but not permanently installed in the system at this time. Start with the defaults and build from there.

**Unity engine notes**

* **Use the provided script (Unity package)** - put it on anything you want to drive the platform
* **Use forces** - you can drive your object with transform.position, or velocity, but forces will give the best results and should be the default until you can make an argument for another option
* **The object with the script must stay alive** to prevent ethernet communication errors (so do not destroy between levels/scenes)

**Show PC specifications**

* Windows 10
* Intel Core i5-6600K 3.5GHz Quad-Core Processor
* EVGA GeForce GTX 1080 8GB FTW Gaming ACX 3.0 Video Card
* Kingston HyperX Fury Black 16GB (2 x 8GB) DDR4-2133 Memory
* Samsung 840 EVO 120GB 2.5" Solid State Drive
* Gigabyte GA-Z170-HD3 ATX LGA1151 Motherboard
* Cooler Master Hyper 212 Evo 120mm Case Fan
* EVGA SuperNOVA NEX 650W 80+ Gold Certified Fully-Modular ATX Power Supply
* Chenbro Chassis, 4U, CEB ATX, 3x5.25", 5x3.5", 7 Slots

**Display Specifications**

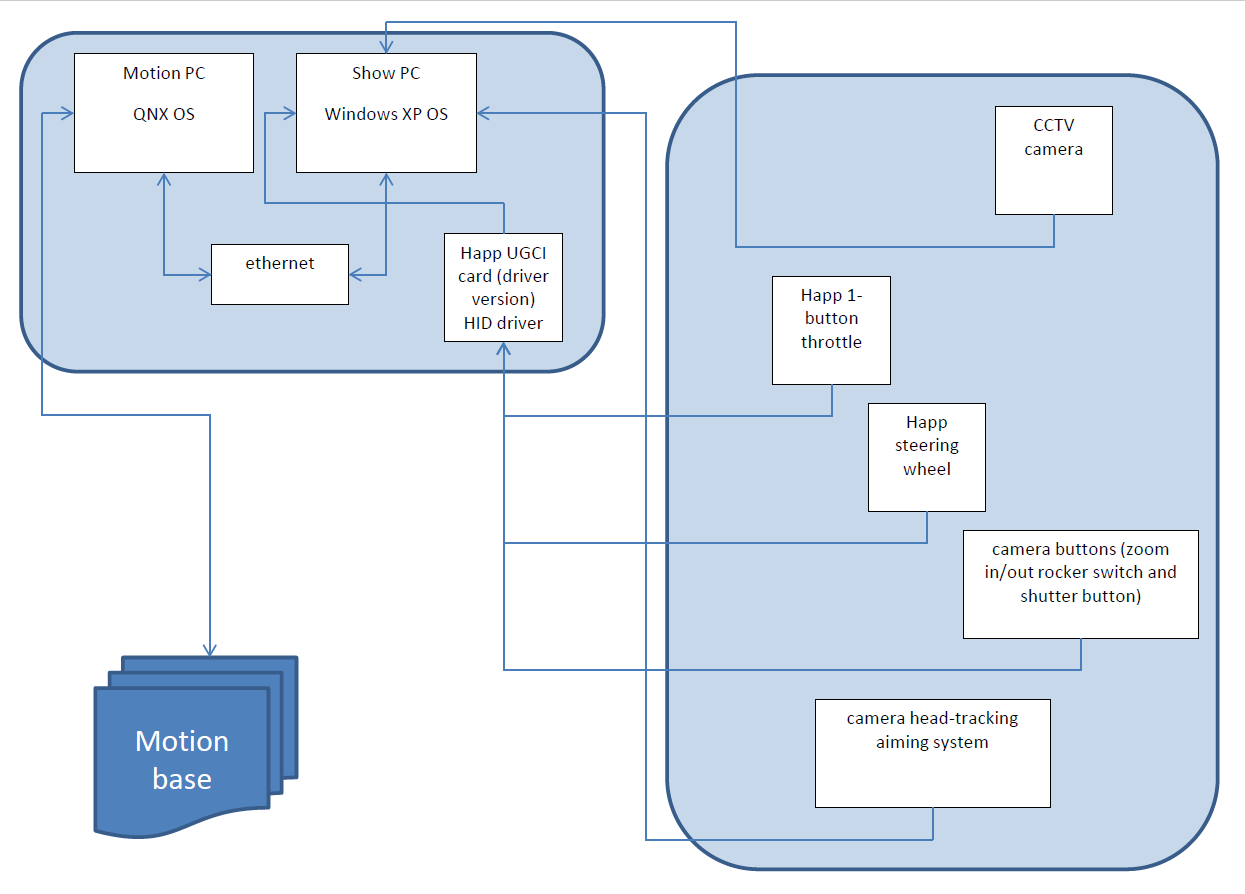
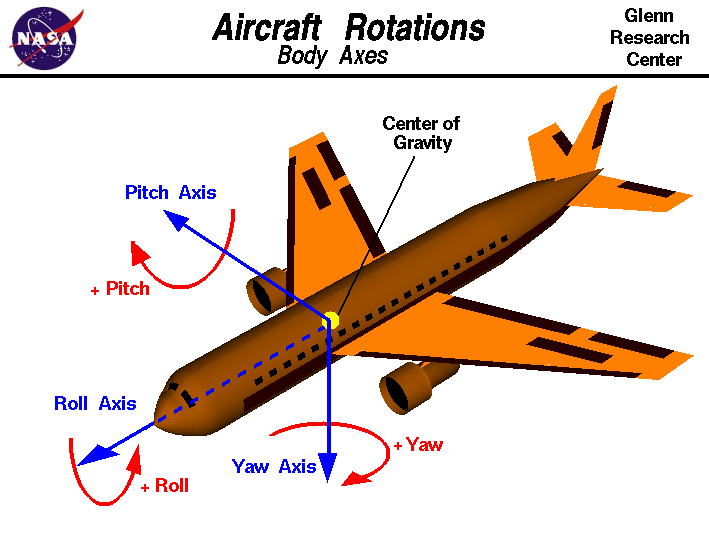
* Front projection
* 1280 x 720 HD resolution

**Final Polish notes**

* We currently use Steam to launch games
* **Create and embed a game icon** in your build .exe.
* **Create and assign a game banner** image for Steam tile/grid view (jpg or png, 460 x 215 resolution)
* **Skip the dialog for final game builds (not for dev builds**). When making a build to run on the motion system, the platform and controls are known so bypass the launch dialog for a streamlined launch. Full screen at 1280 x 720 resolution (16:9 aspect ratio).

**Design specs/guidelines**

****

* **3 degree of freedom (DoF) platform**
  + Roll, +/- (software limited) ~25 degrees
  + Pitch +/- 30-45 degrees (variable – tied to lift position)
  + Lift +/- 12” (24” total travel)
  + 
* **System Architecture**
  + 
* **Guest flow**
  + Riders (~5-minute cycle)
    - ~1.5m load
    - 3-4 minute ride duration
    - ~1.5m unload
  + Queue
    - playable stations spaced through queue OR final spaces in queue
    - guests’ mobile devices
  + Post (survey, swag, sell, shake hands)
* **Rider Coordinate system**
  + We use a variation of aerospace coordinate standards (where Y & Z are swapped, we invert the rotation for them, too – see below this graphic)
  + [](https://www.grc.nasa.gov/www/k-12/airplane/rotations.html)
  + X = (linear) “surge” +forward, -reverse
  + Y = (linear) “lift” +up, -down
  + Z = (linear) “sway” +left, -right
  + X = (rotation) “pitch” +nose up, - nose down
  + Y = (rotation) “yaw” +left, -right
  + Z = (rotation) “roll” +left, -right
  + *FYI, rotations get* [*different labels*](https://www.mathworks.com/help/aeroblks/about-aerospace-coordinate-systems.html) *in other systems.*