AI Grand Prix Virtual Qualifier Technical Specification

Document ID: VADR-TS-002
Issue: 00.02
Date: 2026-05-08

Table of Contents

1. Document Control 3
1.1 Revision History 3
1.2 Audience 3

2. Purpose and Scope 4
2.1 Purpose 4
2.2 Scope 4
2.3 Out of Scope 4

3. Simulation Environment 5
3.1 General Environment 5
3.2 Physical Simulation Model 5
3.3 Spatial Reference Model 5
3.4 Visual Environment 5
3.5 Environmental Determinism 5
3.6 Drone chassis 6
3.7 Gate dimension 6
3.8 Coordinate Frames 7

4. Communication Protocol — MAVLink Interface 8
4.1 Overview 8
4.2 Transport 8
4.3 Supported MAVLink Messages 8
4.4 Timing 8
4.5 Telemetry 8
4.6 Vision Stream 9
4.7 Software-in-the-Loop Bridge 9

5. Contestant Software Environment 10
5.1 Runtime Environment 10
5.2 Client Responsibilities 10
5.3 Intended Control Architecture 10

6. Example Control Session 11

7. Compliance 11

8. Round One — Qualification Phase 11
8.1 Objective 11
8.2 Course Structure 11
8.3. Maximum Run Duration 11

1. Document Control

1.1 Revision History

Issue	Date	Author	Summary of Changes
00.01	2026-03-09	KH	First release
00.02	2026-05-04	NT	camera

1.2 Audience

This specification is addressed to competition participants (“Teams”). It defines the technical interface and operational requirements required to develop autonomous control software for the Virtual AI Drone Race.

2. Purpose and Scope

2.1 Purpose

This document defines the interface between contestant-developed control software and race simulator. The specification ensures that all teams interact with the simulator through a consistent communication and control interface.

2.2 Scope

- communication interfaces between contestant software and the simulator
- control input requirements
- telemetry interfaces
- vision data interfaces
- simulation timing and performance constraints
- virtual race environment definition
- qualification run requirements

2.3 Out of Scope

- internal simulator architecture
- event operations
- contractual or commercial matters

3. Simulation Environment

3.1 General Environment

The race takes place within a high-fidelity real-time physics simulator.

- start gate
- sequential race gates
- finish gate
- vertical and horizontal obstacles
- boundary elements
- terrain and environmental structures

Gates will be visually distinctive to the environment, but consistent throughout the Virtual Qualifier 1 track.

3.2 Physical Simulation Model

The simulator implements a rigid-body drone flight model including thrust generation, aerodynamic drag, gravity, and collision physics.

Physics update frequency: 120 Hz.

3.3 Spatial Reference Model

The simulator uses a local Cartesian coordinate system internally. No geographic coordinates are provided. GPS simulation is not available and absolute global position is not exposed.

3.4 Visual Environment

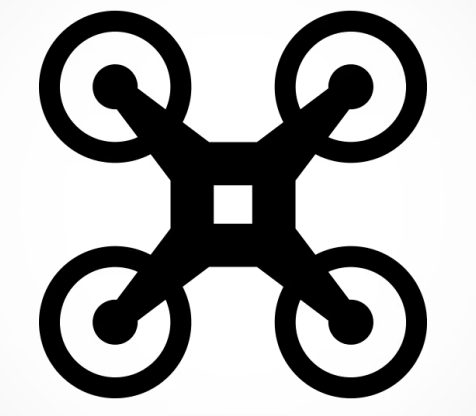
The simulator provides a forward-facing first-person camera and visual environment including gates, course guidance structures, static scene objects, and dynamic lighting.

3.5 Environmental Determinism

- course geometry is identical for all participants
- physics parameters are identical
- environmental conditions are deterministic

3.6 Drone chassis

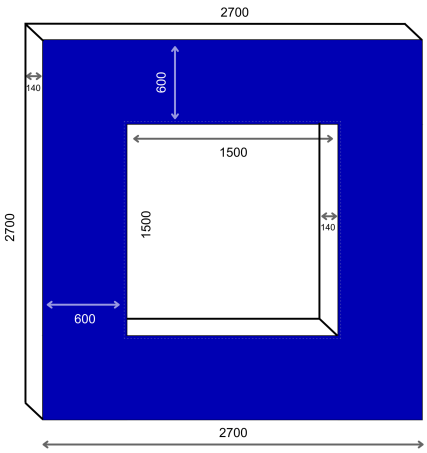
- Width: 280mm
- Length: 280mm
- Height: 160mm



3.7 Gate dimension

Gate boundaries

- Width: 2700mm
- Height: 2700mm
- Depth: 260mm



Gate inner square boundaries:

- Width: 1500mm
- Height: 1500mm
- Depth: 260mm

3.8 Coordinate Frames

The Mavlink2 coordinate convention is NED

Frame Constant	Description
MAV_FRAME_LOCAL_NED	The origin (0,0,0) is a fixed physical point on the ground (usually where the drone armed).
MAV_FRAME_BODY_NED	The origin is the vehicle itself. X points forward, Y points right, Z points down.

Body to Camera

The camera and the body frame the same origin. The camera is tilted upwards by 20° upwards. Be aware that all coordinates are NED and you might need to rotate the camera frame into the camera coordinate convention of your specific image processing library.

Body to IMU

The body to imu transformation is the identity map.

Camera intrinsics

We use a standard pinhole camera model without lens distortion.

- Image resolution: 640px x 360 px
- [cx,cy] = [320px,180px]
- [fx,fy] =[320,320]
- VFoV= 90°

4. Communication Protocol — MAVLink Interface

4.1 Overview

simulator communicates with contestant control software using https://github.com/mavlink/c_library_v2 through MAVSDK-compatible interfaces.

4.2 Transport

Supported transport: UDP

4.3 Supported MAVLink Messages

Please refer to the MAVLINK 2 documentation for the definition of messages
https://mavlink.io/en/guide/mavlink_2.html

Message	Direction	Purpose
HEARTBEAT	Simulator → Client	Connection status
ATTITUDE	Simulator → Client	Vehicle attitude
HIGHRES_IMU	Simulator → Client	Vehicle status
SET_POSITION_TARGET_LOCAL_NED	Client → Simulator	Control interface
SET_ATTITUDE_TARGET	Client → Simulator	Control interface
TIMESYNC	Simulator → Client	Timing
HIGHRES_IMU	Simulator → Client	Measurements

4.4 Timing

Physics simulation rate: 120 Hz
Command rate <100Hz
Minimum heartbeat rate: 2 Hz

4.5 Telemetry

- vehicle attitude
- orientation
- linear velocities
- system status flags

4.6 Vision Stream

Frequency of the camera stream is 30 HZ with a resolution of 640 by 360 pixels.

Transport Summary

- **Protocol:** UDP
- **Port:** `5600` (Default)
- **Byte Order:** Little-Endian (<)
- **Header Size:** 24 Bytes

Packet Structure

Each packet consists of a fixed-length **Metadata Header** immediately followed by a variable-length **Binary Payload**.

Field	Type	Size	Description
frame_id	`uint32`	4B	Unique sequence ID for the image frame.
chunk_id	`uint16`	2B	The index of this packet within the frame (0 to `total_chunks - 1`).
total_chunks	`uint16`	2B	Total number of packets required to complete this frame.
jpeg_size	`uint32`	4B	Total size of the final reconstructed JPEG file in bytes.
payload_size	`uint32`	4B	Size of the JPEG data slice contained in *this* packet.
sim_time_ns	`uint64`	8B	Simulation epoch timestamp in nanoseconds.

4.7 Software-in-the-Loop Bridge

The simulator provides a low-latency UDP SITL bridge enabling external AI controllers to exchange telemetry and control commands.

5. Contestant Software Environment

5.1 Runtime Environment

Participants may assume a Python-based runtime environment. Python 3.14.2 is known to operate correctly. Participants are allowed to choose other environments

The DCL Simulator software runs on windows 11 with a standard PC and a descent GPU with 8GB VRAM

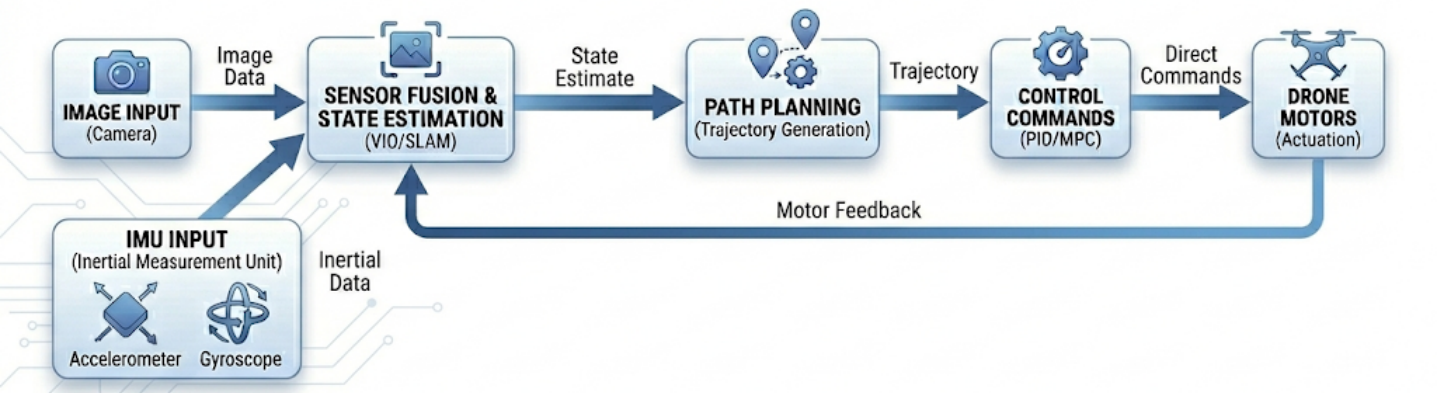
Currently we do not support Linux OS.

5.2 Client Responsibilities

- establish MAVLink communication
- maintain heartbeat messages
- send control commands
- process telemetry data
- process vision stream data

5.3 Intended Control Architecture

Typical conceptual control pipeline:
Vision + Telemetry → Perception → Planning → Control → Pilot Commands → Stabilized Controller



6. Example Control Session

- Client initializes MAVSDK
- Client connects to simulator endpoint
- Simulator transmits HEARTBEAT
- Client streams control commands
- Simulator applies commands
- Telemetry and vision streams returned

7. Compliance

Participants must ensure their implementation conforms to this specification. Specifically, human interaction during the flight which the participants submit as a timed run is grounds for immediate disqualification.

8. Round One — Qualification Phase

8.1 Objective

Round One verifies that contestant software can successfully navigate the racecourse.

8.2 Course Structure

- start gate
- intermediate gates
- finish gate

8.3 Maximum Run Duration

Maximum run duration: 8 minutes.