AmesDT: Digital Twin and Autonomy Validation Environment

Rory Lipkis[1], Ignacio López-Francos[2], Pavlo Vlastos[2], and Adrian Agogino[1]

[1] NASA Ames Research Center, Moffett Field, CA

[2] KBR Inc., NASA Ames Research Center, Moffett Field, CA

Background

- Within NASA, many R&D efforts involve integrating ML models into autonomous systems
  - Vision-based control and navigation
- V&V capabilities are lagging this effort
  - Cannot be validated like traditional HW/SW components
  - Tolerances and risk assessments less meaningful
  - Failure modes harder to describe and analyze
- ML training often requires simulation
  - Lowers cost but further complicates V&V

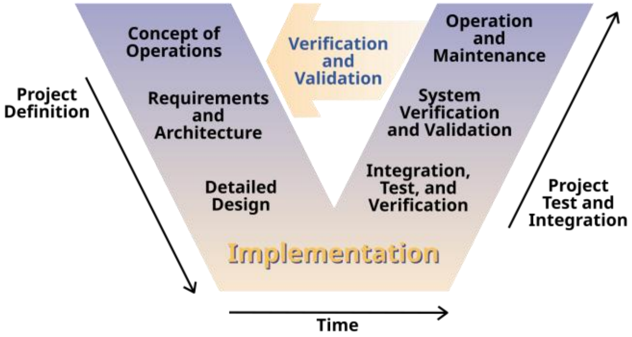


Image source: Wikipedia / V-model

Motivation

We need to characterize how technologies developed in simulation- or model-based environments can be transferred effectively to the real world.

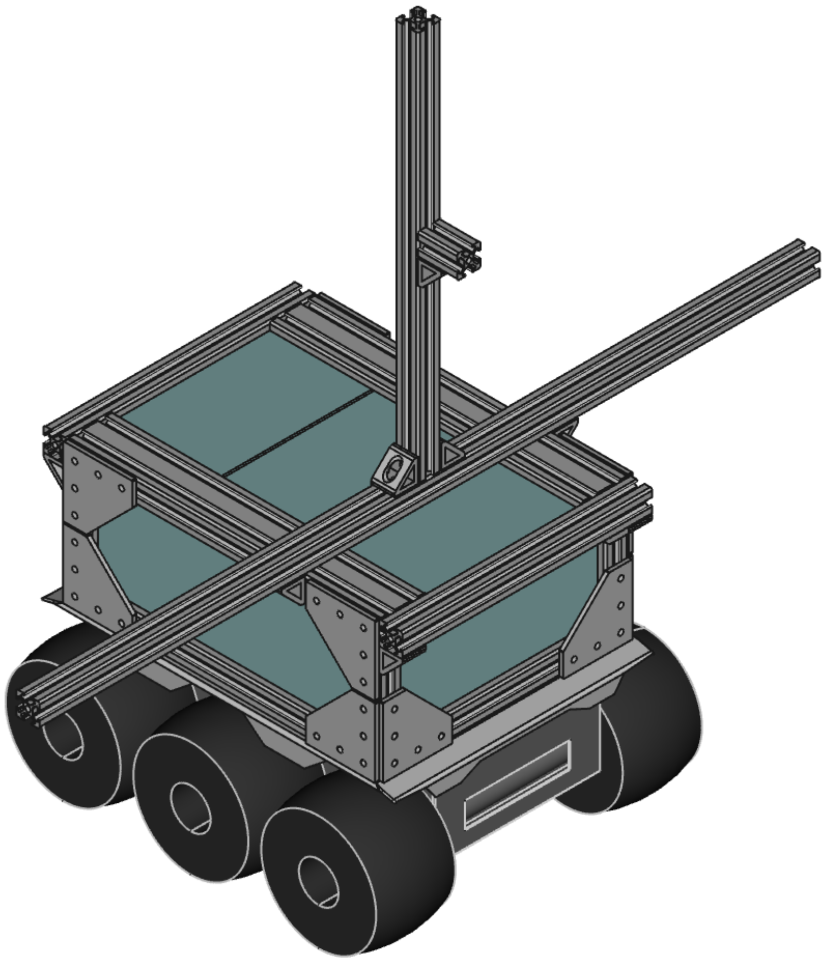
- Digital twins are a compelling framework for real-time diagnostics and prognostics in aerospace.
- Solution: create a *different* sort of digital twin, specialized for the investigation of ML components and their generalization characteristics.

NASA Ames Digital Twin (AmesDT)

- Simulation of research vehicle at NASA ARC campus
  - Navigable physical environment
  - Rover dynamics
  - Camera input
  - Autonomy software stack
- Corresponds to deployable physical twin
- Allows research into V&V for ML models developed in simulation
  - Standard digital twin: discrepancies are reduced with data.
  - AmesDT: discrepancies *create* the data.

AmesDT requirements

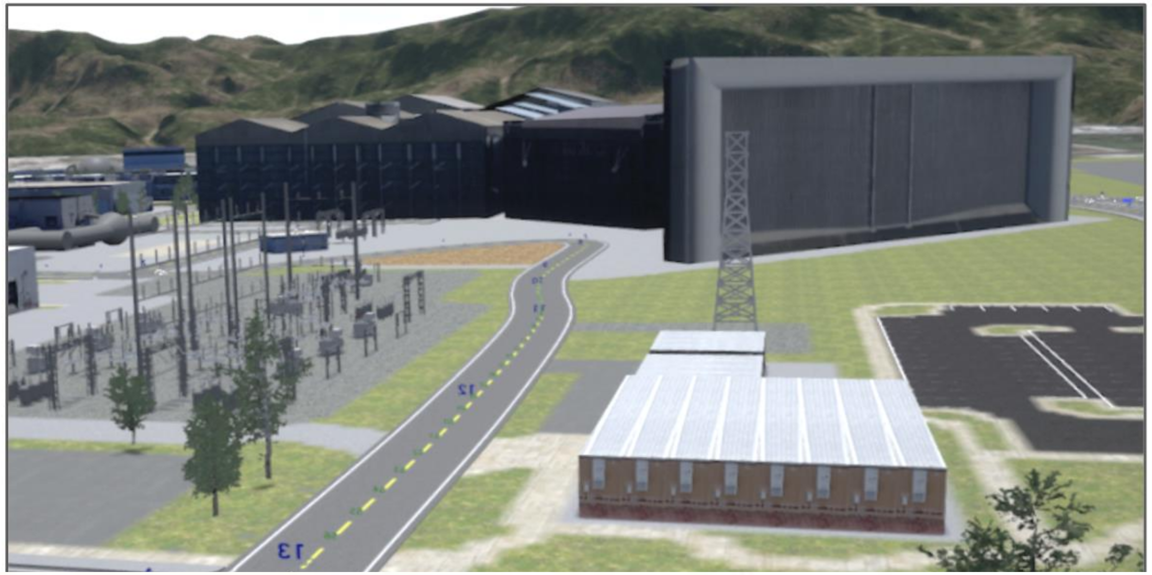
- Rough correspondence to real world
- Adequate visual fidelity
- Adjustable environmental conditions
- Modifiable vehicle dynamics
- Programmable control and I/O interfaces
- Full determinism
- Faster-than-real-time execution



Development process

- Unreal Engine 4 (Epic Games)
  - Simulation base, environment / object creation utilities, rendering
  - Custom patches: accelerated execution
- AirSim (Microsoft)
  - Physics, weather effects, image segmentation, control interfaces
  - Custom patches: rover dynamics, determinism, additional sensors
- Satellite imagery
  - Building reconstruction and layout, road geometry, foliage, groundcover

Examples of usage



Challenges

- Heavily GUI-based workflow
- Physics “plausibility” ≠ accuracy
- Overcomplexity of full-fledged video game engine for specialized simulation
- Scalability of development process
    - Manual mapping of satellite imagery to buildings

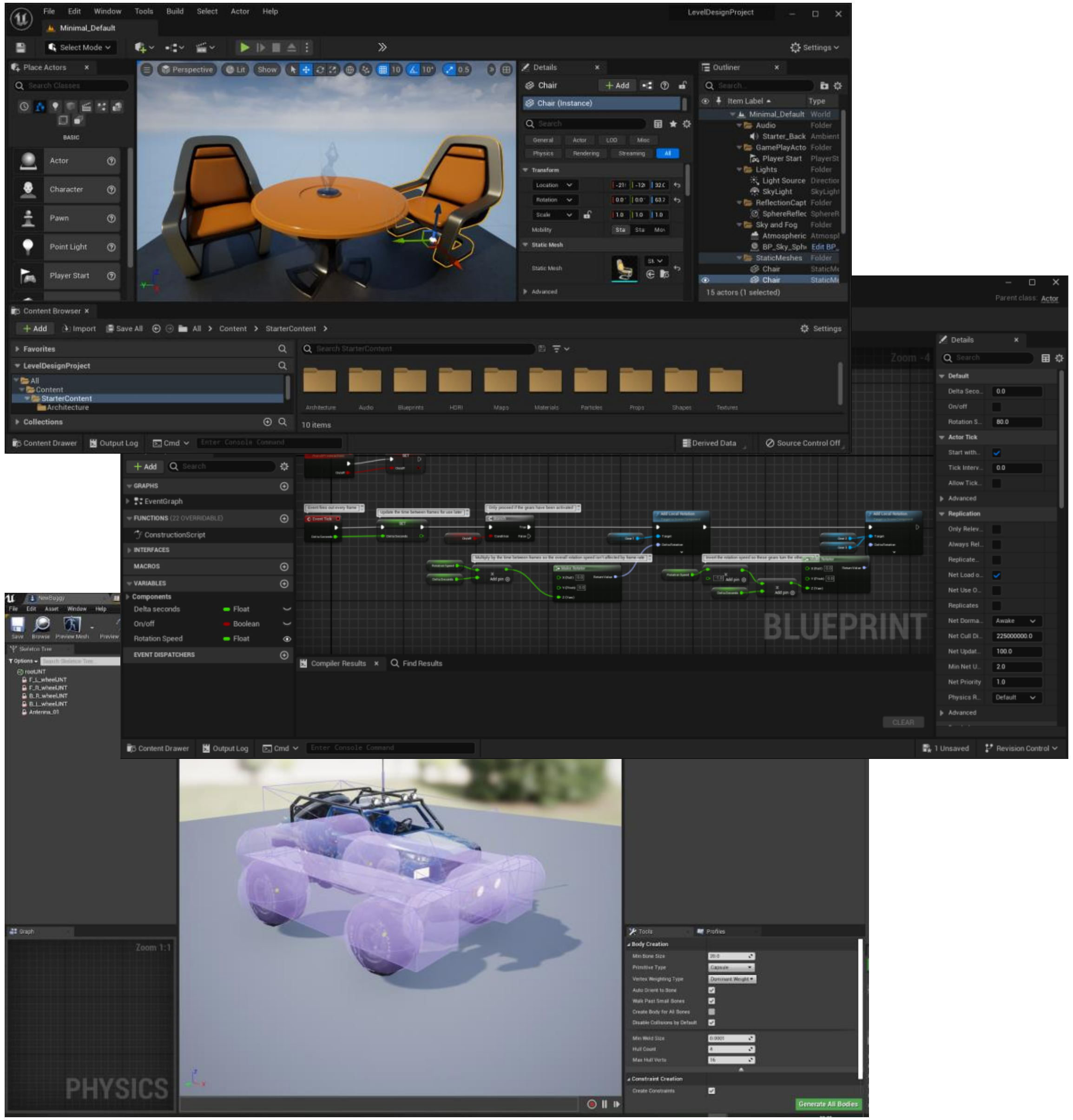


Image source: dev.epicgames.com

Questions | Comments

rory.lipkis@nasa.gov