Game Engine-based Agent Simulator (GEAS)

Alexander B. Strout

Advisor: Roy M. Turner

Why use a simulator?

- Autonomous Vehicles (AVs) require control software
- Control software requires a testing environment
  - Ideally, the vehicle itself
- However, this might not be best
  - There may be limited quantity of vehicles
  - May require additional knowledge to operate
  - Possibly slow iteration time
  - May take vehicle away from more productive work
  - Experimental work could possibly damage vehicle

Existing simulators

- CADCON[Chappell et al.], SMART[Turner et al.], DIS-Java-VRML[Brutzman et al.]

- However, these have their own problems
  - Typically highly specialized for their vehicle
    - Control software likely unique to the vehicle
  - Typically built from ground up
    - Rendering, physics, etc.
  - Often under-maintained; can quickly become obsolete

Alternative?

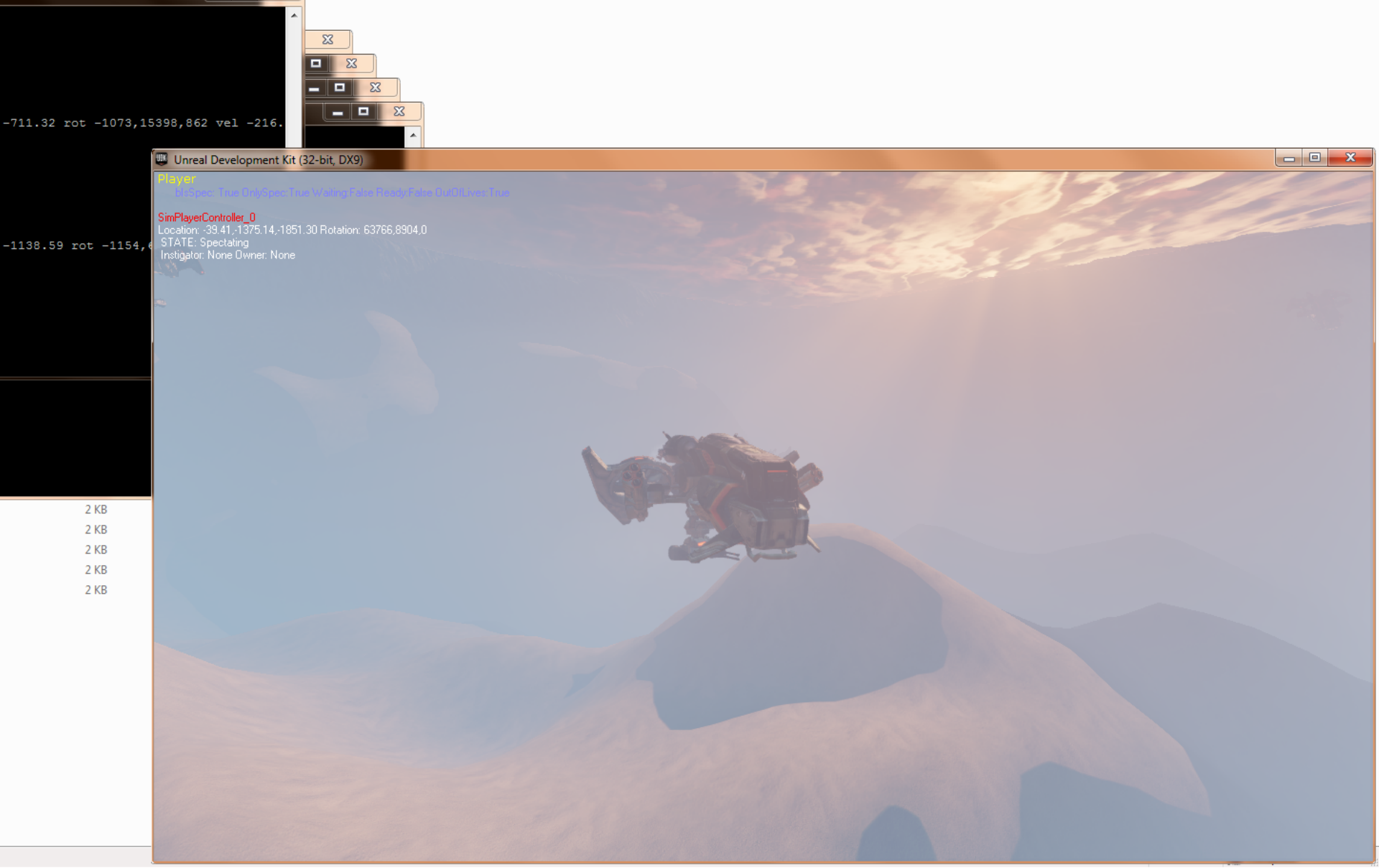
- Game Engines
  - CryEngine, Unity3D, Unreal, Polycode, Torque3D
  - Small to large-scale game development
- Eliminate need to “reinvent the wheel”
  - Rendering
  - Physics
  - Content creation tools
- Consistently maintained and updated
- Free (or reasonable commercial licenses)

“Serious Games”

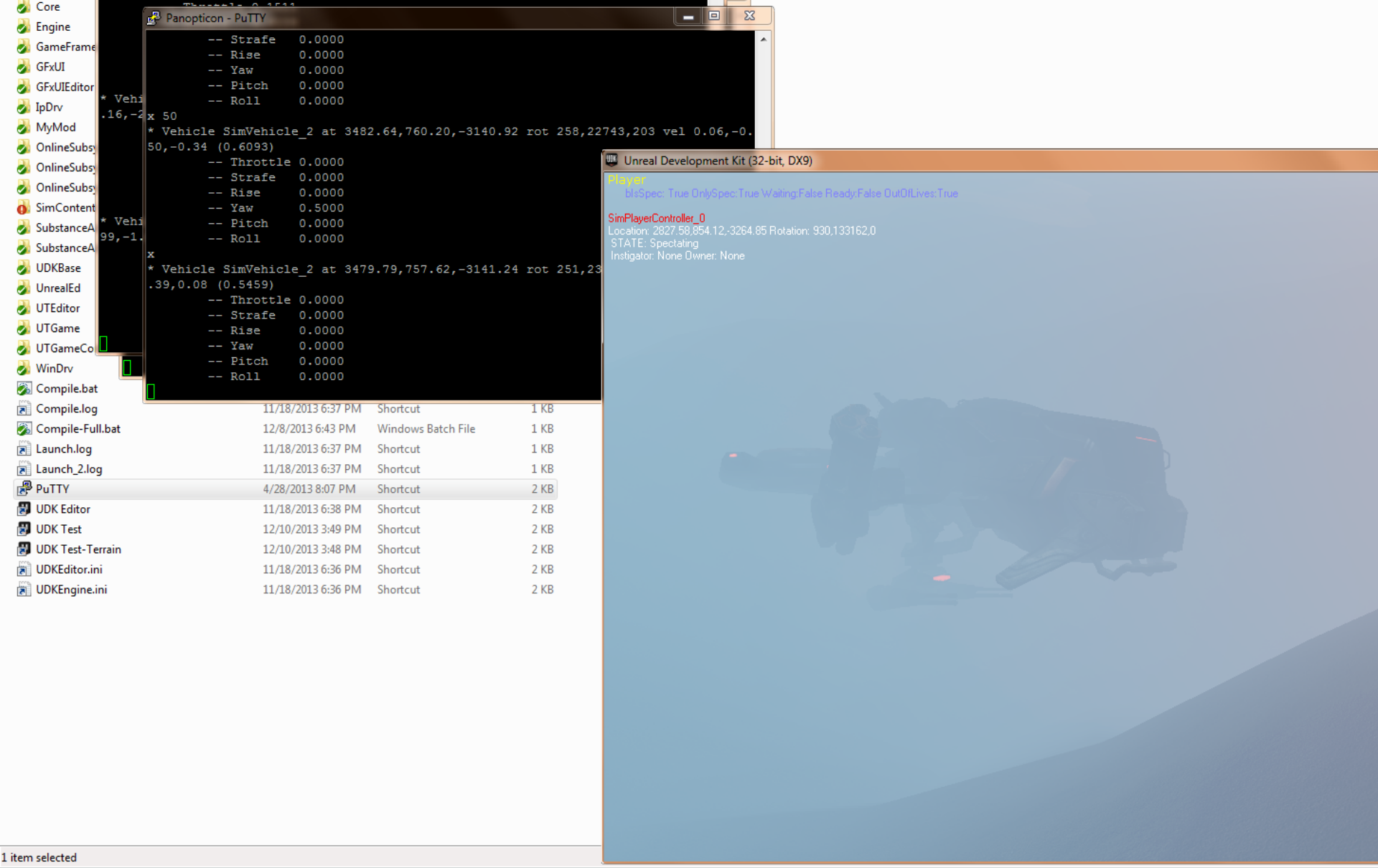
- Already used for military, educational, training or medical purposes
  - VBS[BI Studios], DARWARS[BBN]
  - Flight Simulator[Microsoft]
  - Re-Mission[HopeLab]
- But very little for AV development
  - IAAVS[Periera, Rossetti] - but geared toward traffic simulation
- Need some way to work with control software

Game Engine-based Agent Simulator (GEAS)

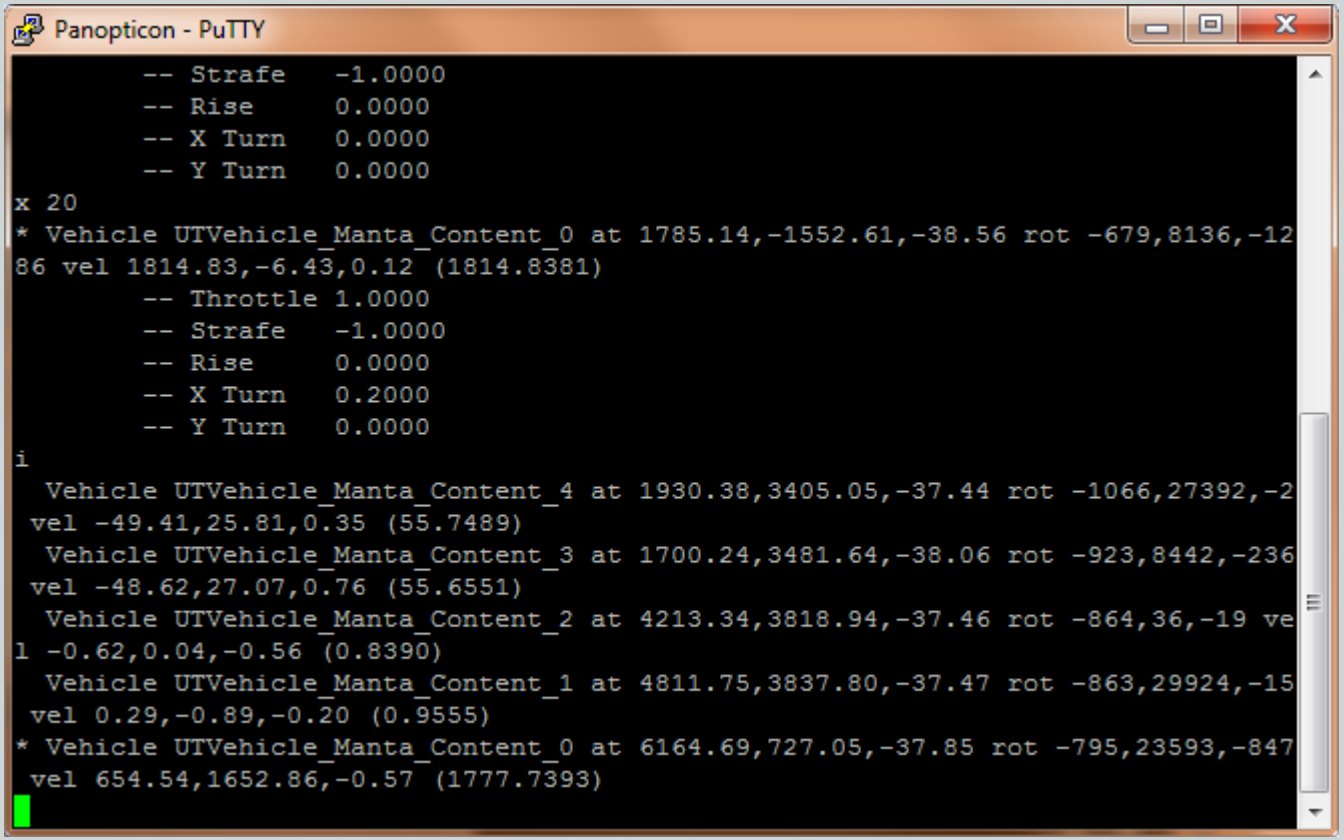
- Unreal and Java
- High-level AV simulation
- Flexible
- Extendible
- Portable



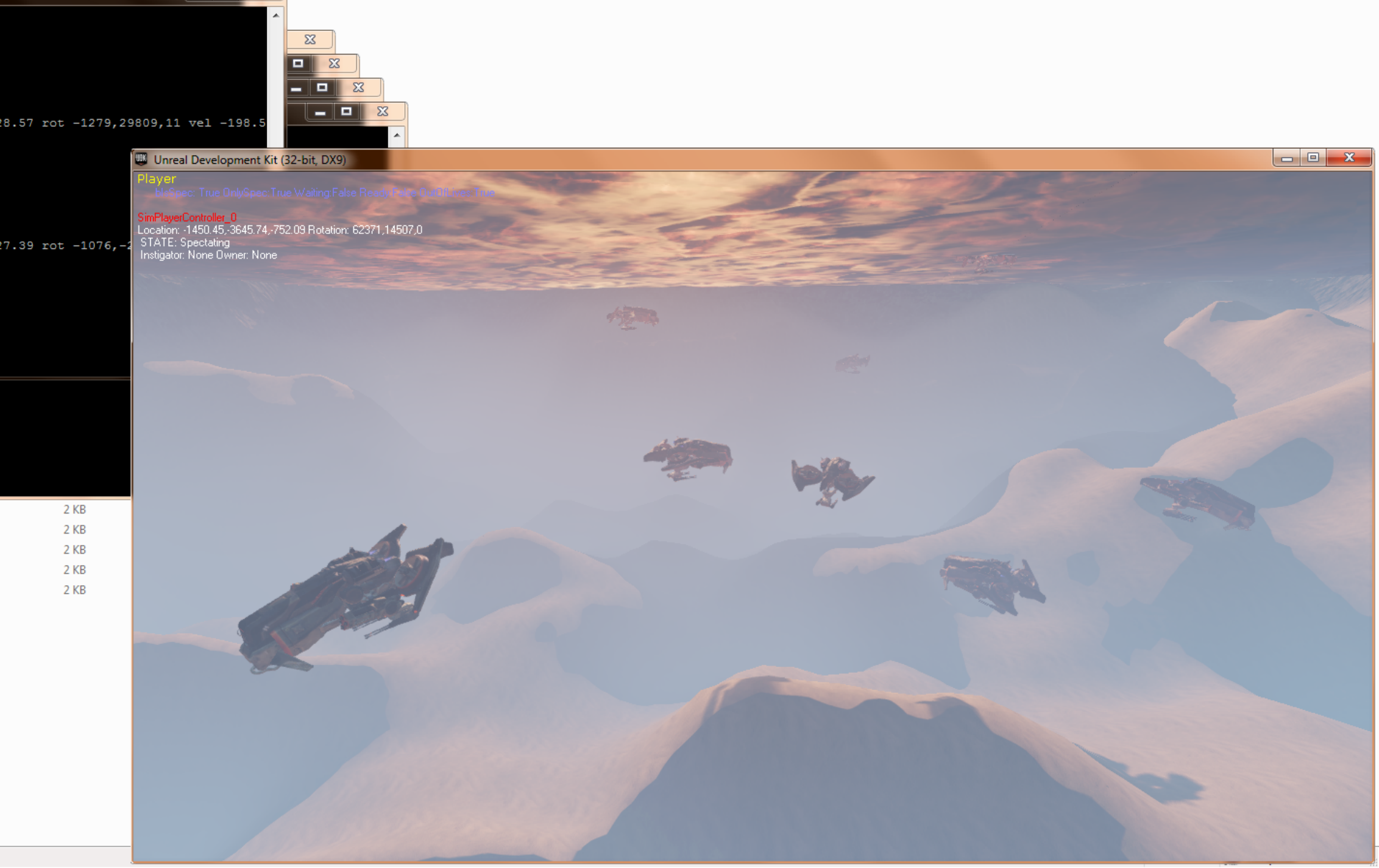
- Autonomous Underwater Vehicles (AUVs)
  - But designed to be flexible for any type of vehicle!



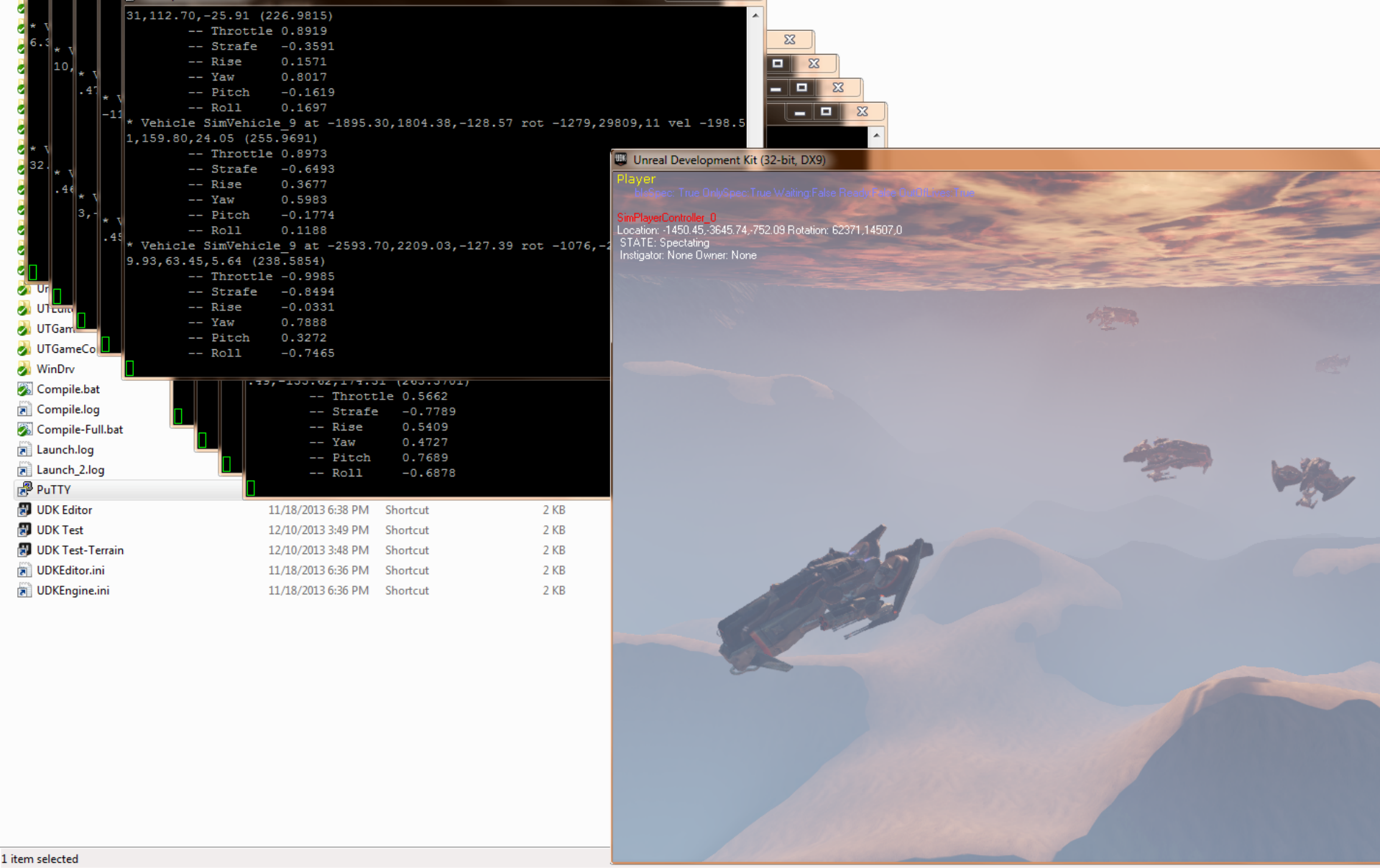
- Simulated vehicle *not* controlled in-game, but instead by external controller via socket connection



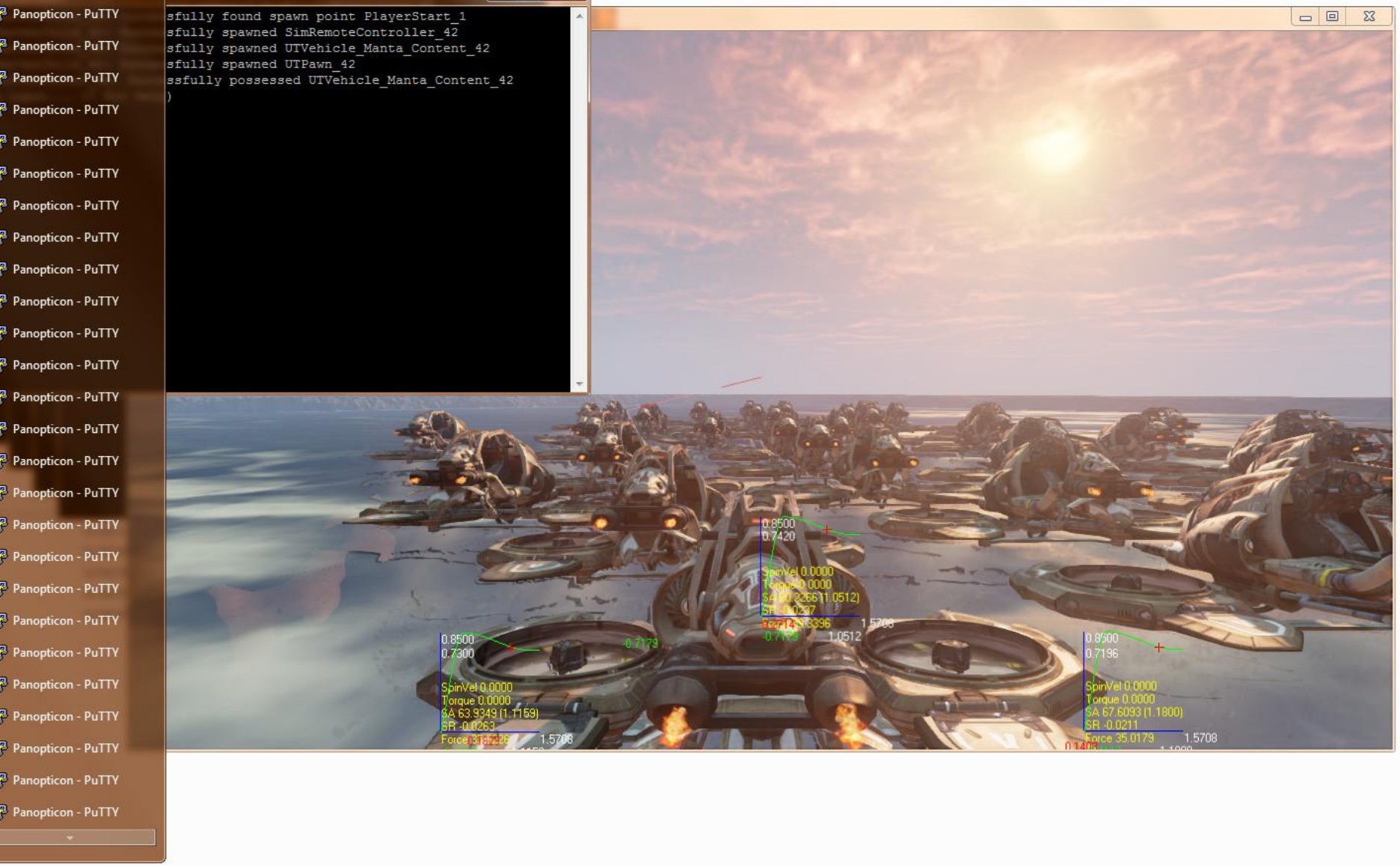
- Simulated sensor data to external controller
  - “GPS”, Depth, Lidar, etc.



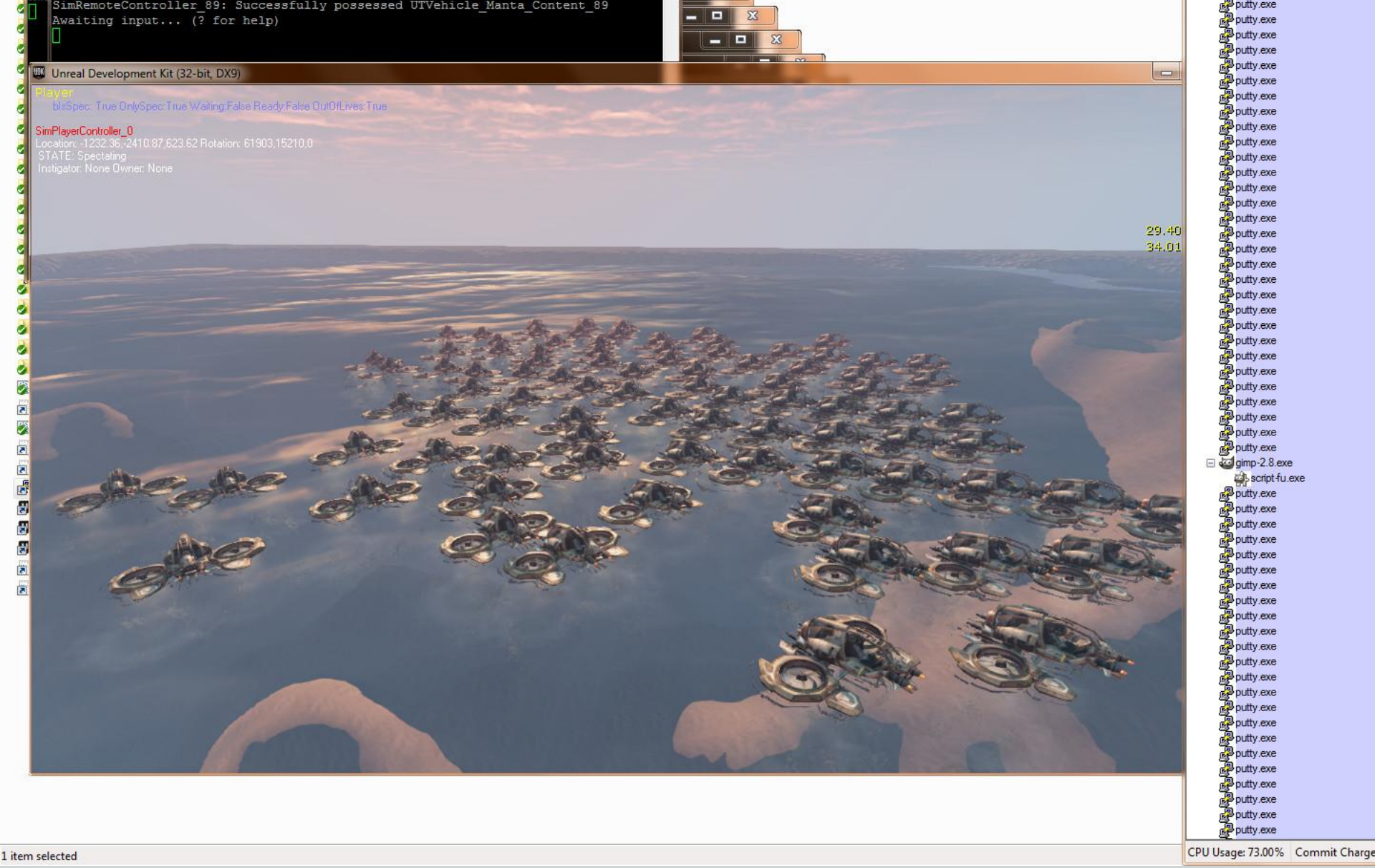
- Supports simulation of multiple agents...



- … each bound to their own controller

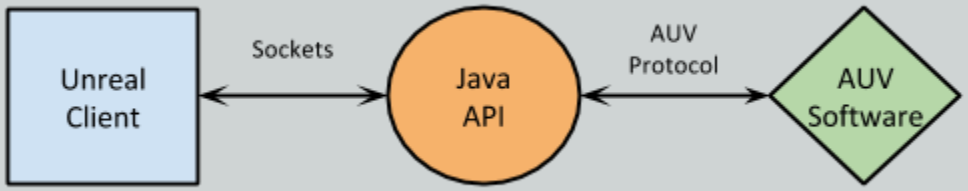


- Highly scalable...



- … up through dozens, hundreds, or possibly thousands of simultaneous agents

GEAS Structure

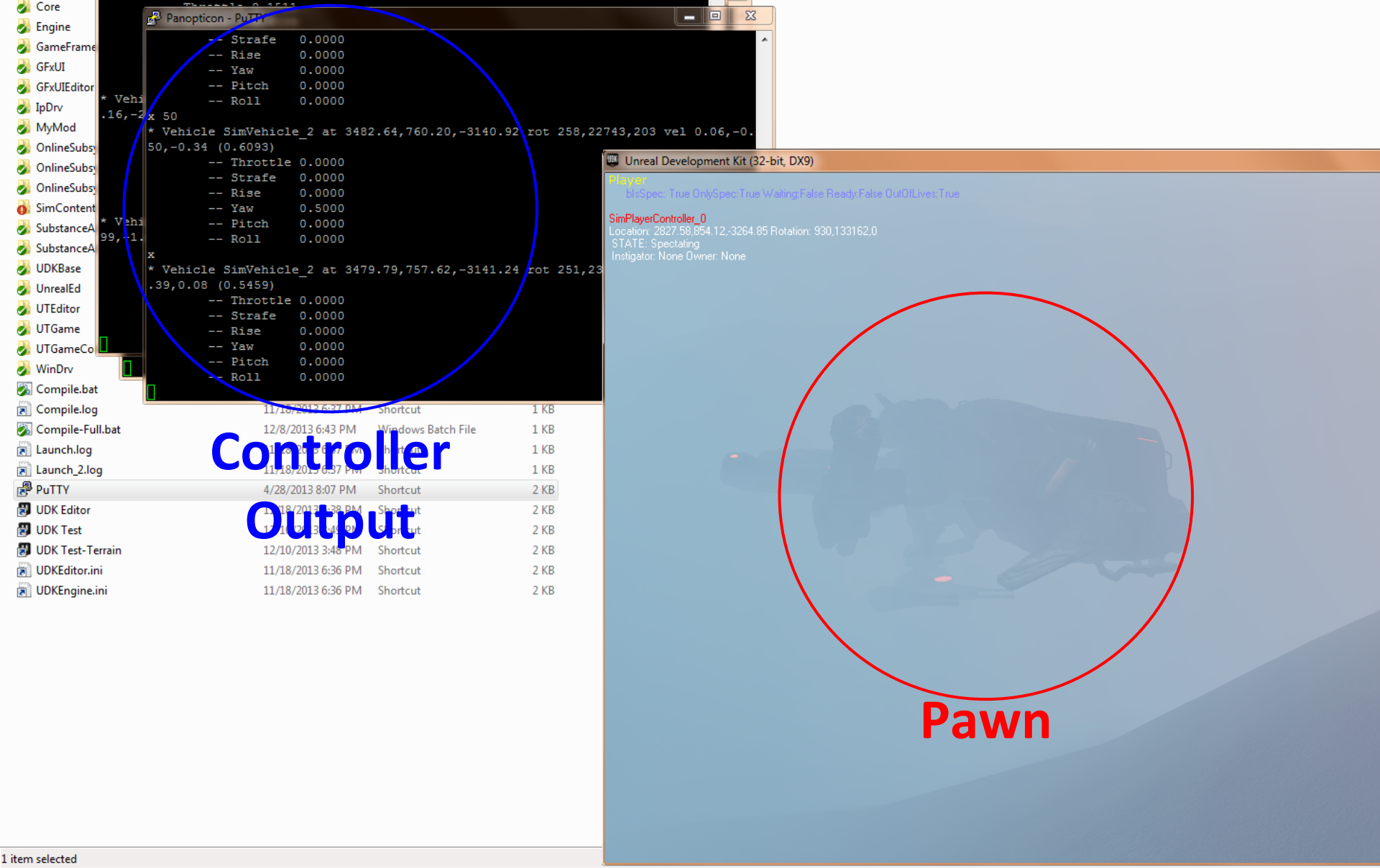


- Unreal Client
  - Game world, on-screen objects
- Java API
  - Optional interface to help AUV software utilize I/O
- AUV Software
  - Controls in-game objects based on perceived data

GEAS: Unreal - Structure

- UnrealScript
  - Object-oriented
  - Used for game logic (vs. underlying engine logic)

- Object
  - Actor
    - Pawn
      - “Phyiscal” properties - movement, collision
    - Controller
      - “Drives” pawn



- Pawn: AUV
- Controller: Output over socket

GEAS: Unreal - Input

- Input over socket maps directly to in-game input
- Movement
  - Throttle
  - Strafe
  - Rise (Elevation)
- Rotation
  - Pitch
  - Yaw
  - Roll

GEAS: Unreal - Sensor Emulation

- “GPS”
  - Position
  - Velocity
- Depth
  - Distance from water surface
- Lidar
  - Situational awareness
  - Tracking other objects

GEAS: Unreal - Sensor Emulation - “GPS”

- Position and Velocity as Euclidean vectors
  - <X, Y, Z> world coordinates
    - e.g. 127, -34, -5816
  - <X, Y, Z> velocity
    - Can resolve X, Y, Z components into actual velocity

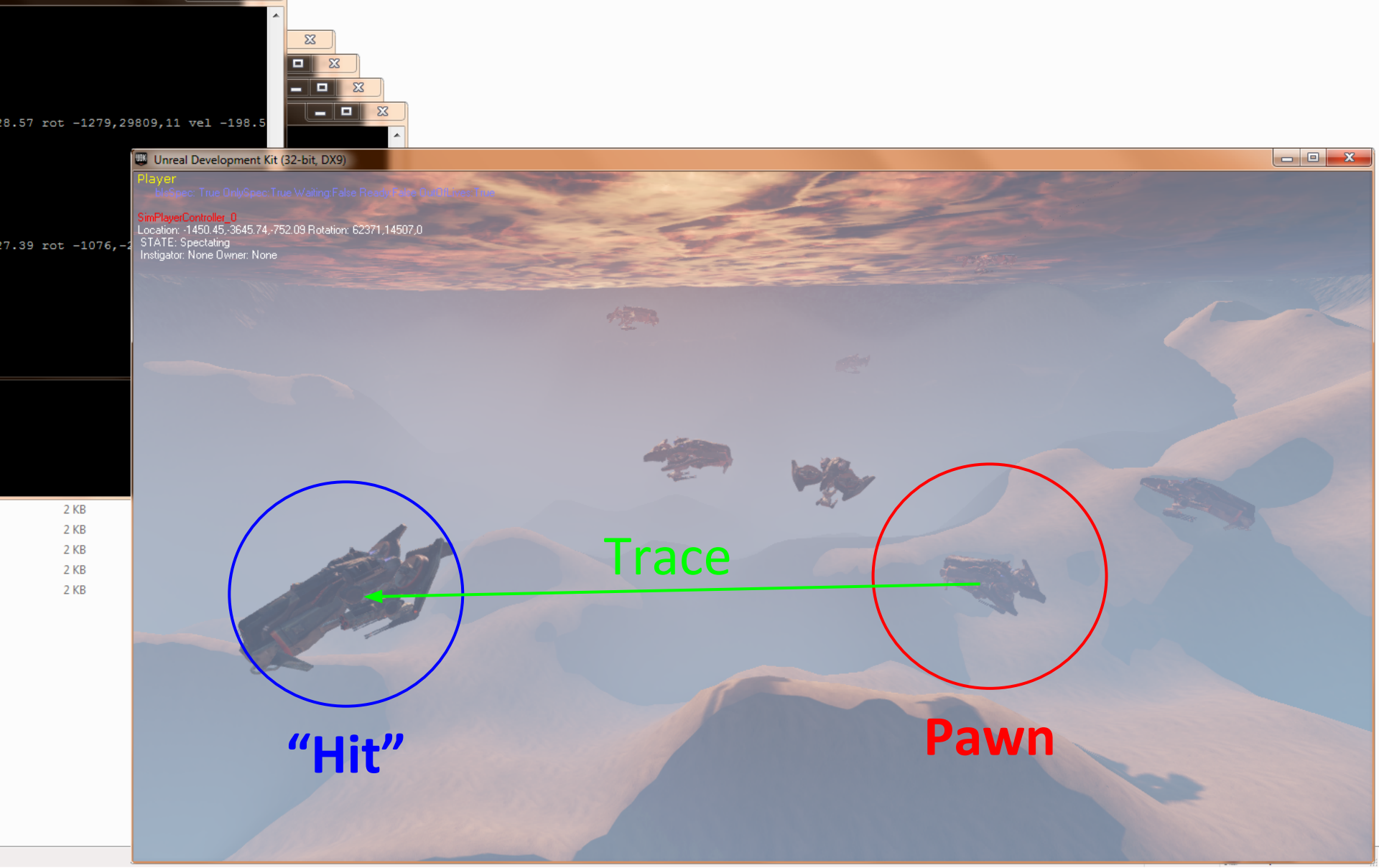
- In “Unreal Units”
  - no relation to real-world units - virtual world!

GEAS: Unreal - Sensor Emulation - Depth

- Water surface at elevation “0”, so…
- Simply Z-component of world position

GEAS: Unreal - Sensor Emulation - Lidar

- Traces from AUV Pawn
  - Fire “bullet”, see what it “hits”
- Returns <X, Y, Z> “hit” position
  - Distance by comparing to AUV Pawn position
- Also returns “hit” object, if any
  - Can examine object velocity, orientation, etc.



- Trace from AUV that "hits" another AUV

Future Work

- Fleshed out simulation
  - More accurate hydrodynamics
    - Temperature, pressure, currents, etc.
  - Interactive objects (fish, coral reefs, seaweed, etc.)
  - Additional sensor modules (Sonar etc.)
  - Sensor noise models
- Extended Java API
  - Built-in protocol support, etc.
- Distribution to research community as extendible framework

Conclusion

- Three-part autonomous vehicle simulation
  - In-game 3D simulation
  - Game <-> Controller interface
  - External AI controller
- Scalable and portable
  - Unreal and Java for a variety of hardware
- Extendible
  - Decoupled systems easy to repurpose

More Info

Strout, Alexander B., Turner, Roy M. (2013).

A Game Engine-based Simulator for Autonomous Underwater Vehicles,

18th International Symposium on Unmanned Untethered Submersible Technology (UUST),

Portsmouth, NH, August 11--14, 2013