Building Simulation and Control in Unity 3D

Jonathan Fürst, Gabe Fierro, Philippe Bonnet, David E. Culler
jonf@itu.dk, gt.fierro@berkeley.edu

Context
We are abstracting physical building infrastructure including Devices, Sensors and Actuators to a form of building operating system \[2\]. On top of it, applications, e.g., a Building Automation System (BAS) like OpenBAS are been run.

Unity 3D
- Multi-platform Game Engine
- C#, JavaScript, Boo.
- Rendering, Physics, Lighting
- Import model via Autodesk Revit
- Native & Web Apps, iOS, Android

sMAP [1] Integration
- Virtual sMAP Drivers
- Configuring plug devices, lights, HVACs by associating them with power, heat, cool, light values.
- Read JSON from running sMAP source and auto maps with imported Autodesk Revit Model or in an abstract form if not available.

Future Work
- Use feedback from sensors and actuators to improve model.
- Implement model predictive control (e.g., predict weather every hour and adjust HVAC settings).
- Separate Visualization from simulation and control engine.
- Detailed simulation report.

Problem
We can't just test and debug new configurations or applications:
- How can we test without harming an actual physical building or its people?
- How can a building manager better decide on different schedules and configurations?

Approach
- We need to tightly couple a building control environment with a building simulation environment.
- We need to provide an easy to use interface and visualization.
- Our implementation is using a game engine as execution environment to allow a fast prototyping for visualization, simulation and control.

Process Flow

Hierarchical Structure

References

Our system uses virtual device drivers, providing a feedback to the simulation engine. Output of the simulation engine is then used as input for the control engine. After adequate testing, virtual drivers are switched to drivers actuating physical devices and the output of the simulation engine is thereby substituted by actual sensor data.

Conclusion