PDSim: Simulate Plans

Emanuele De Pellegrin, Ronald P. A. Petrick
Edinburgh Centre for Robotics
Heriot-Watt University
Edinburgh, Scotland, United Kingdom
ed50@hw.ac.uk, R.Petrick@hw.ac.uk

Abstract

This paper presents new additions and improvements to the Planning Domain Simulation (PDSim) project. PDSim is a plugin for the Unity game engine to simulate planning domains and plans. PDSim uses a novel way to translate the output of a planner to 2D or 3D animations and effects. Using these visualisation techniques, PDSim aims to help users evaluate the quality of a plan and improve domain and problem modelling. In this system demonstration, we present the structure and operation of the PDSim interface by illustrating how to set up a new simulation for planning problems, and how the user interface can be used for creating animations.

Introduction

Modelling planning domains and ensuring the accuracy of plans can prove to be difficult, particularly when tackling real-world problems. While plans may be considered valid, relying exclusively on the output of planners may not always enable the identification of mistakes in domain modelling that would become apparent when represented visually, such as through a 2D or 3D visualization (Chen et al. 2020).

Visualisation of planning domains and plan solutions is not a new idea, but an active research topic with many approaches that use visualisation methods to illustrate generated plans (Vrakas and Vlahavas 2005; Vaquero et al. 2007; Chen et al. 2020; Tapia, San Segundo, and Artieda 2015; Muise 2016; Le Bras et al. 2020; Roberts et al. 2021; Shah et al. 2021). All of these systems aim to assist users in understanding how a plan is generated and help detect potential errors in the modelling process.

The Planning Domain Simulation (PDSim) system (De Pellegrin 2020) introduced a new method for visualising and simulating classical planning problems defined in PDDL. PDSim approaches the problem by building a graphical environment for plan visualisation and simulation that harnesses the framework and components of the Unity game engine (Unity Technologies 2020) to deliver a 3D or 2D virtual visualisation of the planning problem. PDSim aims to lower the barrier for using planning technology both in industry and in academia by reducing the knowledge needed to use PDDL, and by gamifying the process of planning. With PDSim, the user is able to customise the model that represents the objects of the planning problem and define animations for such objects using Unity-based features, such as basic transformation of an object (translation, rotation, scale), path following between two points on a map, playing a sound when a particular condition is met, spawning a particle system, among others The user is able to create real-world scenes that reflect the execution environment of the planning problem, exploiting the functionality of the Unity game engine as an enhancement to existing automated planning tools. Later developments have extended the initial tool and implemented a new approach that simplified the process for defining animations using an intuitive visual scripting language (De Pellegrin and Petrick 2021, 2022).

Planning Domain Simulation (PDSim)

PDSim (De Pellegrin 2020; De Pellegrin and Petrick 2021, 2022) has been developed as a plugin for the Unity game engine, a 2D and 3D game environment widely used in the video game industry. Unity offers an intuitive graphical user interface (GUI) editor and many available components, such as a built-in physics engine, realistic shaders and materials, and a path-planning library. Thanks to its modularity, it is possible to extend the interface to fit user needs, for instance, by defining custom animations for PDDL objects or by extending existing animations to reuse models from different simulations without creating them from scratch every time. Figure 1 shows the overall system diagram for PDSim, highlighting the key steps for every new simulation.

A simulation is initialised and handled by the back-end server, which is integrated with the AIPlan4EU's Unified Planning Framework (UPF) (Micheli et al.),[1] a recent addition to PDSim which is responsible for parsing and building a JSON representation of the planning model. UPF is a planner-agnostic framework for Python, which increases PDSim's modularity and lets users select their preferred planner implementation, separating it from the simulation stage itself which is handled later in the process. PDDL files are translated into a JSON map of the components needed for simulation. PDSim uses domain components such as the definitions of actions, types, and predicates to set up the core simulation in Unity. The user can also use the Unity editor to

[1]https://github.com/aiplan4eu/unified-planning

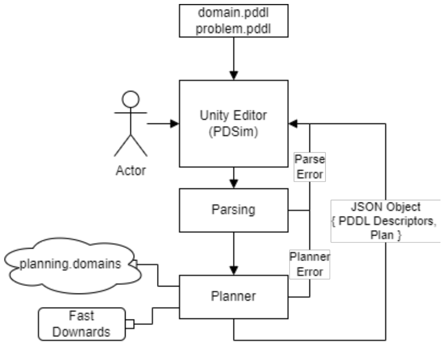

Figure 1: PDSim System Diagram

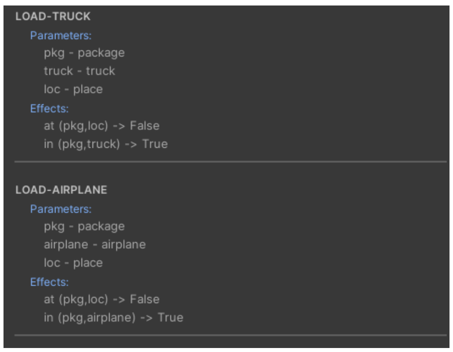

Figure 2: PDDL actions in PDSim from JSON

configure multiple problems for the same domain, and thus multiple simulations for different plans. For instance, Figure 2 shows an example of how actions are translated from PDDL to JSON to C# objects that can be used in PDSim. UPF can also call external planners to generate plans. Recent changes to PDSim's animation system have also aimed to improve the ease in which animations and simulated objects are defined and customised. For instance, Figure 3 shows PDSim's new animation system which use Unity's Visual scripting. Using the system, animations can be created by adding nodes and connecting animations between them.

In addition to demonstrating the basic PDSim system, the demo will also present how PDSim can operate with custom languages to simulate changes in an environment state. Figure 4 shows how PDSim's predicate animation system is used in a digital twin environment. In this environment, predicates are defined as sensors in an Internet of Things (IoT) assistive daily living lab. Using PDSim, we can easily simulate plans and changes in the state of objects and robots. During the demonstration, the current state of integration with ROS (Quigley et al. 2009) will also be presented along with a demo of how to use ROSbags to simulate sensors in the IoT environment and how PDSim can be customised for real-world robotics tasks. This will involve setting up Unity animations and recalling them within PDSim. Additionally, all the improved user interfaces containing information displays such as in Figure 5 and input systems will be presented to the user during the live demo.

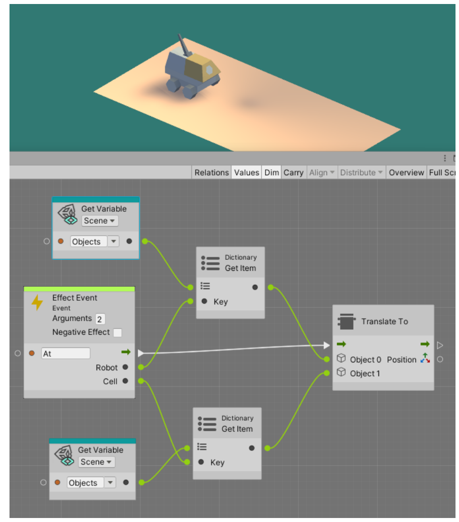

Figure 3: Animation System with Unity's Visual Scripting

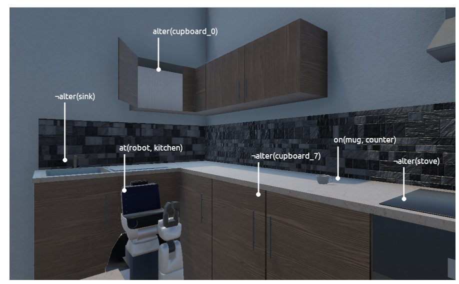

Figure 4: PDSim for real-world robotics: HSR robot (Yamamoto et al. 2019)

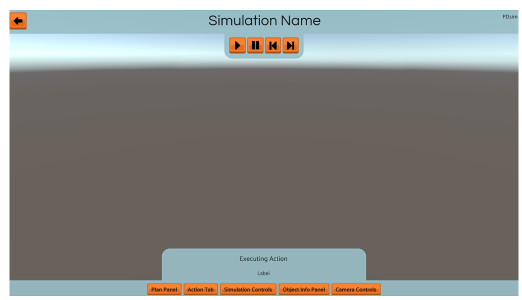

Figure 5: PDSim new User Interface

Conclusion

In this paper, we describe the features of the PDSim system, an extension to the Unity game engine to simulate planning domains and plans. PDSim focuses on user interaction to specify the animations and models relative to a given PDDL domain and problem. In this system demonstration, we will present the main PDSim front-end interface and back-end integration, the system components used to create a simulation (both for a toy problem and a real-world scenario), how to create custom animations from scratch, and how PDSim communicates with ROS.

References

Chen, G.; Ding, Y.; Edwards, H.; Chau, C. H.; Hou, S.; Johnson, G.; Sharukh Syed, M.; Tang, H.; Wu, Y.; Yan, Y.; Tidhar, G.; and Lipovetzky, N. 2020. Planimation. In *ICAPS System Demonstration System*. ArXiv:2008.04600.

De Pellegrin, E. 2020. PDSim: Planning Domain Simulation with the Unity Game Engine. In *Proceedings of the ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*.

De Pellegrin, E.; and Petrick, R. P. 2021. Automated Planning and Robotics Simulation with PDSim.

De Pellegrin, E.; and Petrick, R. P. 2022. What Plan? Virtual Plan Visualization with PDSim.

Le Bras, P.; Carreno, Y.; Lindsay, A.; Petrick, R. P. A.; and Chantler, M. J. 2020. PlanCurves: An Interface for End-Users to Visualise Multi-Agent Temporal Plans. In *Proceedings of the ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*.

Micheli, A.; Arnold, A.; Bit-Monnot, A.; Bonassi, L.; Framba, L.; Gerevini, E.; Satchi, S. H. S.; Helmert, M.; Ingrand, F.; Kockemann, U.; Lima, O.; Patrizi, F.; Pecora, F.; Poveda, G.; Roger, G.; Saetti, A.; Saffiotti, A.; Scala, E.; Serina, I.; Teichteil-Koenigsbuch, F.; Trapasso, A.; and Traverso, P. ???? Unified Planning: A Python Library Making Planning Technology Accessible.

Muise, C. 2016. Planning.domains. In *ICAPS System Demonstration Session*.

Quigley, M.; Gerkey, B.; Conley, K.; Faust, J.; Foote, T.; Leibs, J.; Berger, E.; Wheeler, R.; and Ng, A. 2009. ROS: an open-source Robot Operating System. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*. Kobe, Japan.

Roberts, J. O.; Mastorakis, G.; Lazaruk, B.; Franco, S.; Stokes, A. A.; and Bernardini, S. 2021. vPlanSim: An Open Source Graphical Interface for the Visualisation and Simulation of AI Systems. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 486–490.

Shah, N.; Verma, P.; Angle, T.; and Srivastava, S. 2021. JEDAI: A System for Skill-Aligned Explainable Robot Planning. In *Proceedings of the Demonstration Track, International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. ArXiv:2111.00585.

Tapia, C.; San Segundo, P.; and Artieda, J. 2015. A PDDL-based simulation system. In *Proceedings of the International Conference on Intelligent Systems and Agents (ISA)*.

Unity Technologies. 2020. Unity.

Vaquero, T. S.; Romero, V.; Tonidandel, F.; and Silva, J. R. 2007. itSIMPLE2.0: An Integrated Tool for Designing Planning Domains. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 336–343.

Vrakas, D.; and Vlahavas, I. 2005. A Visualization Environment for Planning. *International Journal of Artificial Intelligence Tools*, 14(6): 975–998.

Yamamoto, T.; Terada, K.; Ochiai, A.; Saito, F.; Asahara, Y.; and Murase, K. 2019. Development of human support robot as the research platform of a domestic mobile manipulator. *ROBOMECH journal*, 6(1): 1–15.