T-World: a flexible, portable and graphical environment for Artificial Intelligence research and education

Sergio Burdisso, Guillermo Aguirre, and Marcelo Errecalde
Universidad Nacional de San Luis, Ejército de los Andes 950, San Luis - Argentina
sergio.burdisso@gmail.com, gaguirre@unsl.edu.ar, merreca@unsl.edu.ar

Abstract. testbeds have proven to be useful tools for Artificial Intelligence research, particularly when we are interested in studying Intelligent Agents. A well-known testbed is the *Tileworld*, which consists of a simulated robot agent and a simulated environment which is both dynamic and unpredictable. Both the agent and the environment are highly parameterized, enabling one to control certain characteristics of each in order to, for instance, experimentally investigate the behavior of various meta-level reasoning strategies. However, little attention has been devoted to provide implementations of the Tileworld capable of exploiting the full potential this testbed might offer as a tool for research and education. This paper introduces a system we have called T-World, a portable, flexible and graphical platform composed by a testbed, a simulated 3D environment and a Web-based GUI. T-World testbed is based on several improvements of the classic Tileworld and provides a visually appealing cross-platform work environment in which agents can be coded in any programming language.

Keywords: testbed, Tileworld, Artificial Intelligence, AI, AIMA, Intelligent Agent

1. Introduction

The most widely accepted definition [6] of the term “artificial agent” remarks the relevance of the environment to explain what an agent is; the type of environment in which an agent acts is extremely important when we need to perform a characterization of it. It is possible to ensure that the key requirement for an agent to exist is to be embedded in an environment in which to act and perceive. The term *percept* refers to the agent's perceptual inputs at any given instant. An agent's *percept sequence* is the complete history of everything the agent has ever perceived. In general, it is said that an agent's choice of action at any given instant depends on the entire percept sequence observed to date. Thus, it is possible to characterize an agent's behavior as a *agent function* that maps any given percept sequence to an action.

Controlled experimentation[2] is a popular methodology used in scientific research, it allows us to measure and study how different program characteristics and environmental settings impact a given system performance. For instance, in the design of central processing units (CPUs), the designer could make use of tools for controlled experimentation such as *benchmarks* or *testbeds* to find the settings that lead to the best performance. Thus, if the interest is in processing speed as a measure of performance, then matrix multiplication is a good benchmark: Good performance on matrix multiplication problems predicts good performance on the large class of numeric tasks for which the processor is being designed.

An early benchmark task for AI planning programs was the “three-block problem”, first described by Gerald Sussman as part of his PhD research[10]. In the problem, three blocks (labeled A, B, and C) rest on a table. The agent must stack the blocks such that A is atop B, which in turn is atop C. However, it may only move one block at a time.

For instance, if the problem starts as shown in Figure 1(1), we can see that reaching

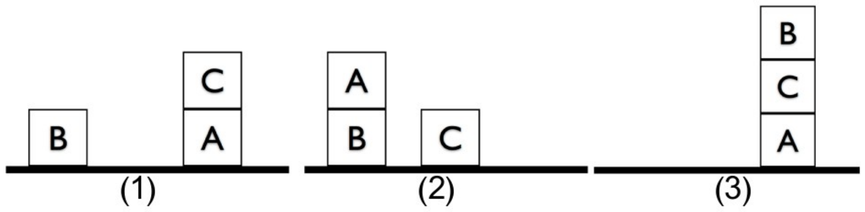

Figure 1. The three-block problem

the goal state is not as simple as separating it into two subgoals: I) get A atop B and II)

get B atop C. Since if we first achieve subgoal I (see Figure 1(2)), to be able to achieve subgoal II the planner needs to undo the already achieved subgoal I, and if instead the planner starts with subgoal II the most efficient solution is to move B. But again, the planner cannot pursue subgoal I without undoing subgoal II (see Figure 1(3)). This simple example exposes what is known as “the Sussman anomaly”, which helped many researchers elucidate how their planners worked. It was popular because like matrix multiplication, it was representative of an important class of problems, those involving interactions among conjunctive subgoals, and it was easy to describe.

Researchers and teachers in the field of artificial agents have long been debating about the relevance of using these kind of tools for controlled experimentation, which help to understand and analyze the phenomena that arise from combining different types of agents and environments. This is not a trivial task, if we take into consideration that this tools should be easy to describe and understand while also being capable of representing most of the challenges that can be found in real world problems. Additionally, such tools should allow the manipulation of different environmental properties such as dynamism, uncertainty, partial observability, etc. so that the problem to be solved by the agent enables the study a wider range of agent types, ranging from a reactive behavior with minimum response times, to a proactive behavior with high-level reasoning skills and capabilities such as long-term planning.

A well-known *testbed* that seems to show many of these desired properties is the *Tileworld*, which has been widely used in research projects allowing researchers the evaluation of several agent architectures and reasoning strategies. Unfortunately, Tileworld as an educational and experimentation tool for students taking courses related to intelligent agents has not had the same popularity and dissemination. From our perspective, one of the reasons for this flaw has been the lack of user-friendly, graphical and interactive environments that meet the graphic standards students commonly found in videogames today. These environments should be portable, easy to use and provide support for teaching as well as all the tools for scientific research that have been traditionally used.

Against this background, the main contribution of this work is to describe and introduce *T-World*, a portable, flexible and graphical platform which in its core consists of a testbed based on several improvements of the classic Tileworld. T-World was developed from scratch with the aim to be used for both, research and educational purposes. With this purpose, the rest of the article is organized as follows. Section 2 describes the classic Tileworld testbed and section 3 describes some experiences with the use of Tileworld in Artificial Intelligence research. Our platform, T-World, is introduced and described in Section 4 remarking how different types of environments can be implemented in this platform. Finally, in Section 5 some conclusions and opportunities of future work are outlined.

2. The Tileworld

The Tileworld[5] was first introduced as a testbed to experimentally evaluate agent architectures. The Tileworld is a chessboard-like grid on which there are agents, tiles, obstacles, and holes[4]. An agent is a unit square which is able to move up, down, left, or right, one cell at a time. A tile is a unit square which behaves like a tile: it slides, and rows of tiles can be pushed by the agent. An obstacle is a group of grid cells which are immovable. A hole is a group of grid cells, each of which can be “filled in" by a tile when the tile is moved on top of the hole cell; the tile and hole cell disappear, leaving a blank cell. If a hole becomes completely filled, the agent gets points for filling it in. The agent knows ahead of time how valuable the hole is; its overall goal is to get as many points as possible by filling in holes. A Tileworld simulation takes place dynamically: it begins in a state which is randomly generated by the simulator according to a set of parameters, and changes continually overtime. Objects (holes, tiles, and obstacles) appear and disappear at rates determined by parameters set by the researcher, while at the same time the agent moves around and pushes tiles into holes. Among these parameters, user is able to set the frequency at which objects (holes, tiles and obstacles) can appear and disappear; also it is possible to set whether holes have either hard timeouts or gradually decay in value. In Tileworld Holes appear in randomly selected empty squares, and exist for a length of time known as their *life expectancy*, unless they disappear prematurely due to the agents actions. The actual time for which a hole exists is its *lifetime*. The interval between the appearance of one hole and the next is known as the *gestation period*. Each hole has a specific value, its score. Life-expectancies, gestation periods, and scores are taken from independent random distributions. The agent performance is measured by running a Tileworld simulation for a predetermined time and measuring the amount of holes successfully filled by the agent. Therefore, the agent's *performance measure* (u) on a given simulation, *r*, is defined as follow:

$$u(r) = \frac{\text{number of holes filled by the agent on r}}{\text{total number of holes on r}}$$

This gives us a normalized performance measure ranging from 0 (i.e. the agent hasn't filled any hole) to 1 (the agent has successfully filled every hole). Beyond its simplicity, the Tileworld enables the study of important capabilities of agents, such as the ability to react to unpredictable changes and to consider new opportunities. For example, suppose an agent is pushing a Tile towards a hole (Figure 2 (a)), when

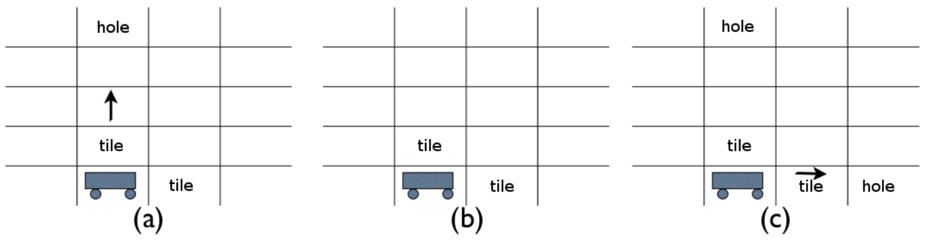

Figure 2. Consideration of opportunities

suddenly the hole disappear (Figure 2 (b)). At this point the agent's goal becomes meaningless and the best the agent could do is, if he is aware of the change, to reschedule its initial goal. To illustrate what "recognize opportunities" would mean in this example, suppose the agent is under the same circumstances as described in Figure 2 (a) and suddenly a hole appears on the right side as shown in Figure 2 (c). It is more likely for the agent to fill in the new hole (since he has to push the tile only one cell) rather than the old one, therefore it is probably better for the agent to change the goal to fill the new just-created hole.

In more detail, the Tileworld provides a set of "knobs" which can be adjusted to control the evolution of a simulation. The main knobs are: a) *Dynamism*, the rate at which new holes appear; b) *Hostility*, the rate at which obstacles appear; c) *Variability of utility*, difference in hole scores; d) *Variability of difficulty*, differences in hole sizes and distances from tiles; and e) *Hard/Soft bounds*, holes having either hard timeouts or gradually decaying in value. By adjusting the knobs, one can allow conditions to vary from something resembling an unconstrained football field to something like a crowded maze, or from a fixed puzzle to constantly changing chaos. For each set of parameter settings, an agent can be tested on tens or hundreds of randomly-generated runs automatically. Agents can be compared by running them on the same set of pseudo-random worlds.

3. Experiences with the use of the Tileworld

In this section two research papers based on the use of Tileworld are briefly reviewed. These papers are of interest since they help us to highlight the manner in which experiments are carried out, how different simulation parameter settings produce an impact on the performance of agents, and how these differences can be graphically monitored by the researcher.

Evaluation of PRS on the Tileworld. In [3], Kenny and Georgeff (K&G) evaluated, among other things, feasibility of developing systems for empirical measurement of agent performance that are stable, sensitive, and capable of revealing the effect of "high-level" agent characteristics such as commitment. K&G establish some useful concepts that also have been included in our research, such as the already defined holes *life-expectancies* and *gestation periods*.

The experimental system was based upon the PRS real-time reasoning system [1] operating within the Tileworld environment. To be able to measure the agent performance over this type of dynamic environment, the experimental parameter that has the most fundamental influence upon agent effectiveness is the rate γ at which world changes. The Tileworld and K&G agents both measure time by an abstract clock. They execute synchronously, with the ratio of their clock rates set by γ. Therefore, the γ value allows the dynamism of the world, as perceived by the agent, to be varied over a wide range.

K&G defined an agent's effectiveness ε to be its score divided by the maximum possible score it could have achieved. This gave a measure of performance that was largely independent of game length, and proved to be stable and reproducible. By plotting ε as a function of γ and analyzing the obtained effectiveness curves they identify three different section: as γ increases, hole *life-expectancies* decrease reciprocally, as measured by the agent's clock. Eventually holes start to disappear before the agent has filled them, and ε drops below 1. Then, this decline in effectiveness has a sudden onset and is initially steep, with ε falling from 0.9 to 0.5 for a factor of 2 increase of γ. Finally, as γ increases further the decline in ε becomes more gradual and eventually asymptotically approaches zero. K&G used this relationship between these two values to characterize different agent types

Markov decision process and BDI on the Tileworld. In [8], Simari and Parsons (S&P) were concerned with decision making in autonomous agents and, in particular, the tradeoff between the optimal solution provided by MDPs and the more tractable approximation provided by the BDI model. To carry out this task they developed BDI agents and a MDP model for the Tileworld domain. S&P adopted the simplified version of the Tileworld created by Martijn Schut [7]. The simplifications to the model are: tiles are omitted, so an agent can score points simply by moving to a hole; agents have perfect, zero-cost knowledge of the state of the world; and agents build correct and complete plans for visiting a single hole (they do not plan tours for visiting more than one hole).

The BDI agents for the Tileworld are implemented as follows: the agent's beliefs consist of its perceptions of the locations of holes in the world. Various parameters dictate how accurate these beliefs were: *accessibility* (how many positions the agent can see from where it is standing), *dynamism* (how many steps the world takes for each step of the agent), *planning cost* (how many steps the agent must spend in order to build a plan), and the agent's *intention reconsideration* strategy. This last parameter is one of the most important because it has a great influence on how efficient the agent will be; if intentions are reconsidered too often or too soon, this could lead to a waste of effort.

For an MDP model, the world is modeled by taking into account every possible action in every possible state. For the simplified Tileworld, this means that for a world of size n (that is, an $n \times n$ grid) there is a set of 8 actions, n^2 possible positions for the agent, and 2^{n^2} possible configurations of holes. This last number is obtained by considering that every position in the grid may contain a hole or not. A state in this world consists of a pair (P, H), where $P = (i, j)$, $0 \leq i, j \leq n - 1$, and H represents a given configuration of holes on the grid. Thus, the total number of states in this case is $n^2 2^{n^2}$. These values show that even for a very small Tileworld, the MDP model requires an intractable amount of resources in order to compute an optimal policy.

Comparing these approaches in a parameterized fashion, making use of the Tileworld knobs, they concluded that for small worlds, the MDP approach—effectively giving an agent a complete conditional plan—outperforms the BDI approach even if the agent has a full set of linear plans that it switches between. However, when the size of the world becomes larger, the performance of the MDP approximations becomes poorer, and the heuristic nature of the BDI planner is able to deal with the increase in world size without difficulty.

4. The T-World platform

The design and development of T-World (`tworld-ai.com`), within our research group, initially arose from the need for a Tileworld version capable of dealing with both, education and research. Therefore, in order for T-world to meet these requirements, it had to be *portable* and *flexible* to allow researchers and students to use it independently of any programming languages, operating system, frameworks or libraries —that is, users should code their agent programs in any programming language and platform they please. Additionally, aiming to encourage students to use the tool, T-World had to be a visually appealing, user-friendly, graphical and interactive environment and met the graphic standards students commonly found in videogames today.

In T-World, simulations take place in a three dimensional environment (see Figure 3), objects and robots act and move in a realistic and continuous (not discrete) fashion; For instance, users are able to watch the reactions of robots when colliding with obstacles or pushing tiles, tiles falling and filling the holes, etc.

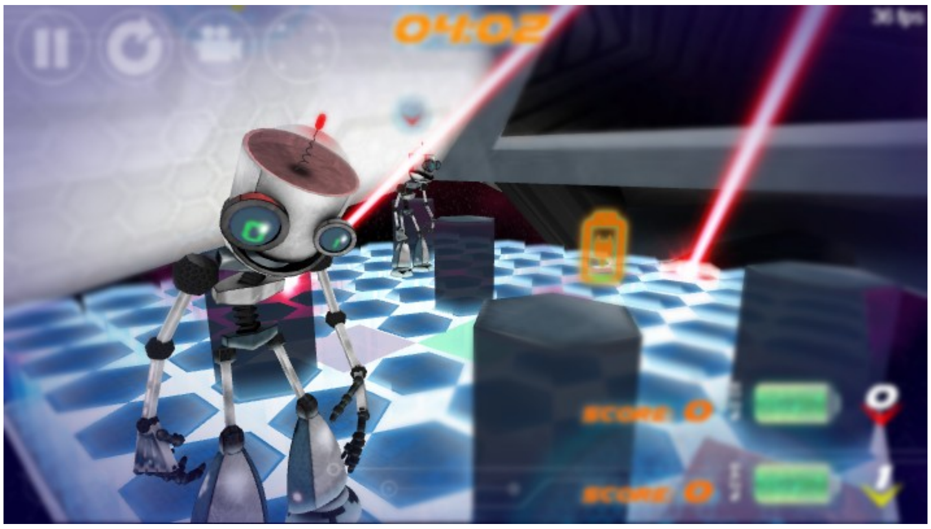

Figure 3. Snapshot of a T-World simulation

The entire T-World logic was completely coded in Javascript since web browsers do not natively support any other programming language. JavaScript is a high level, dynamic, untyped, and interpreted programming language. Alongside HTML and CSS, it is one of the three essential technologies of World Wide Web content production; the majority of websites employ it and it is supported by all modern web browsers *without plug-ins*. JavaScript is prototype-based with first-class functions, making it a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles.

The most common use of JavaScript is to add client-side behavior to HTML pages, which is also known as Dynamic HTML (DHTML). Scripts are embedded in or included from HTML pages and interact with the Document Object Model (DOM) of the page. Some simple examples of this usage are animation of page elements or the creation of interactive content, like games. Since JavaScript code can run locally in a user's browser, the browser can respond to user actions quickly, making an application more responsive. Furthermore, JavaScript code can detect user actions that HTML alone cannot, such as individual keystrokes. In order to run JavaScript code, web browsers make use of what is known as a JavaScript engine, which is is an interpreter that interprets the source code and executes the script accordingly.

4.1. Environment configuration

Given that this paper focuses on the potential use of T-World in Artificial Intelligence education as a teaching tool, this subsection describes the different types of environments that can be created using our tool. This types of environment are categorized along the dimensions introduced in the leading textbook in Artificial Intelligence [6]: observability, number of agents, determinism, dynamism and knowledge of the environment. Users are able to choose and set these parameters through GUI components such as sliders and spinners.

Fully observable vs. partially observable. If an agent's sensors give it access to the complete state of the environment at each point in time, then it is said that the task environment is fully observable. Fully observable environments are convenient because the agent need not maintain any internal state to keep track of the world. An environment might be partially observable because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data. In T-World partially observable environment can be configured using two different approaches: a) O*bservability bound*, number of cells robots are allowed to see around; b) *Noise generator*, percentage of noise while perceiving obstacles, tiles and holes.

Single agent vs. multiagent. As discussed in [6], in some cases *may* be convenient to view a given entity as an agent, and there are other situations in which entities *must* be viewed as agents. In case user needs a task environment in which multiple agents (robots) are going to perform in, T-World provides three different options: a) *competitive multiagent environment*, every robot in the simulation has its own score and they all have to compete against each other in order to obtain the highest score; b) *cooperative multiagent environment*, all robots in the simulation share the same score, so they all have to coordinate actions in order to maximize the score points; c) *competitive and cooperative multiagent environment*; there are multiple teams of robots, all team members share the same score (i.e. each team has its own score) and different teams must compete against each other.

Deterministic vs. stochastic. If the next state of the environment is completely determined by the current state and the action executed by the agent, then it is said the environment is deterministic; otherwise, it is stochastic. In principle, an agent need not worry about uncertainty in a fully observable, deterministic environment. If the environment is partially observable, however, then it could appear to be stochastic. Most real situations are so complex that it is impossible to keep track of all the unobserved aspects; for practical purposes, they should be treated as stochastic. It is said an environment is *uncertain* if it is not fully observable or not deterministic. In T-World, users are able to create stochastic environments by means of a user-defined *stochastic model of actions*, in which the user specify all possible outcomes of an intended action and quantify the uncertainty about outcomes in terms of probabilities; Additionally, the user can choose among a set of predefined *stochastic models of actions* or manually define a new one.

Static vs. dynamic. If the environment can change while an agent is deliberating, then it is said the environment is dynamic for that agent; otherwise, it is static. Static environments are easy to deal with because the agent need not keep looking at the world while it is deciding on an action, nor need it worry about the passage of time. Dynamic environments, on the other hand. are continuously asking the agent what it wants to do; if it hasn't decided yet. that counts as deciding to do nothing. If the environment itself does not change with the passage of time but the agent's performance score does, then we say the environment is *semidynamic*. In T-world is possible to choose any of these three options. Additionally, if the task environment is static, T-World provides a world editor in which the user can graphically draw the initial state.

Known vs unknown. Strictly speaking, this distinction refers not to the environment itself but to the agent's or designer's) state of knowledge about the "laws of physics" of the environment. In a known environment, the outcomes (or outcome probabilities if the environment is stochastic) for all actions are given. Obviously, if the environment is unknown, the agent will have to learn how it works in order to make good decisions. Note that the distinction between known and unknown environments is not the same as the one between fully and partially observable environments. It is quite possible for a known environment to be partially observable. Conversely, an unknown environment can be fully observable.

4.1. Agent program

In T-World all agents perceive the environment in the same manner, percepts are sent as data messages formatted in either JSON, XML or Prolog fact[1] depending on what the user needs are. The percept message carries three types of information: a) *external to the agent*, such as the location of all objects and other agents in the world, time elapsed, etc; b) *relative to the agent itsef*, such as energy levels, score, current location, etc; and c) *built-in knowledge*, those fixed values previously set by the user to control the evolution of the simulation, such as probabilities of actions outcomes, dimensions of the grid, costs of actions, etc.

Thus, users need only to code their agents reasoning engines (also known as *agent programs*) in any programming language, receiving the agent's percepts message from T-World via a socket. Note that the fact that percepts are sent to the agent programs using a standard format, gives T-World the property of being flexible, since all popular programming language support JSON and XML.

1 The JSON and XML formats are open standard data-interchange format that uses human-readable text to transmit data objects consisting of attribute–value pairs and are supported by all popular programming languages.

5. Conclusions and future work

Taking into consideration the important role of testbeds in the study and evaluation of different types of intelligent agents and the important role we believe these tools could play in Artificial Intelligence education as teaching tools, this article proposes, introduces and briefly describes the T-World platform. In order to make T-World portable, we decided to implemented as a web page so that anyone were able to use it via Internet through a web browser (`tworld-ai.com`). T-World is flexible, agent programs can be coded in any programming language since percept are sent using a standard format, such as JSON and XML. Finally, with the aim of encouraging students to use the tool, T-World was developed to be visually appealing, user-friendly, graphical and interactive platform in which simulations take place in a three dimensional world.

Among the immediate uses of the platform, we will integrate T-World in different projects within our research group [9] and with regard to education, we are considering the use of the platform in the computer science degree course, "Artificial Intelligence".

References

1. Michael P Georgeff and Francois Felix Ingrand. Decision-making in an embedded reasoning system. In Proceedings of the 11th international joint conference on Artificial intelligence-Volume 2, pages 972–978. MKP Inc., 1989.
2. Steve Hanks, Martha E. Pollack, and Paul R. Cohen. Benchmarks, test beds, controlled experimentation, and the design of agent architectures. AI Magazine, 14(4):17–42, 1993.
3. D Kinny and M George. Commitment and efectiveness of situated agents. In IJCAI-91, pages 82–88, 1991.
4. Michael Lees. A history of the tileworld agent testbed. Technical report, School of Computer Science and Inf. Tech, University of Nottingham. UK, 2002.
5. Martha E. Pollack and Marc Ringuette. Introducing the tileworld: Experimentally evaluating agent architectures. In In Proceedings of the Eighth National Conference on Artificial Intelligence, pages 183–189, 1990.
6. Stuart J. Russell and Peter Norvig. Artificial Intelligence - A Modern Approach (3. internat. ed.). Pearson Education, 2010.
7. Martijn Schut and Michael Wooldridge. Intention reconsideration in complex environments. In Proceedings of the fourth international conference on Autonomous agents, pages 209–216. ACM, 2000.
8. Gerardo I Simari and Simon D Parsons. Markov Decision Processes and the Belief-Desire-Intention Model: Bridging the Gap for Autonomous Agents. Springer, 2011.
9. Cecilia Sosa-Toranzo, Marcelo Errecalde, and Edgardo Ferretti. On the use of agreement technologies for multi-criteria decision making within a BDI agent. In 14th Ibero-American Conference on Artificial Intelligence (IBERAMIA), November 2014.

10. Gerald Jay Sussman. A Computer Model of Skill Acquisition. PhD thesis, Artificial intelligence lab, MIT, 1973.