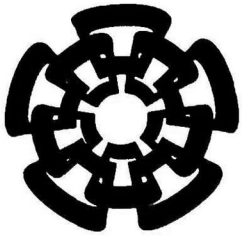
XX(147077.1)



Centro de Investigación y de Estudios Avanzados
del I.P.N.

Unidad Guadalajara

Animación de Criaturas Virtuales Usando Aprendizaje

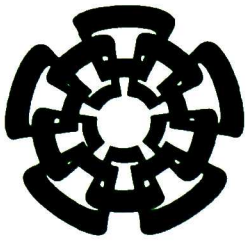
Tesis que presenta:
Moisés Uc Cetina

para obtener el grado de:
Maestro en Ciencias

en la especialidad de:
Ingeniería Eléctrica

Director de Tesis:
Dr. Félix Francisco Ramos Corchado

Guadalajara, Jalisco, Mayo de 2008.



Centro de Investigación y de Estudios Avanzados
del I.P.N.

Unidad Guadalajara

Animation of Virtual Creatures Using Learning

A thesis presented by:
Moisés Uc Cetina

to obtain the degree of:
Master in Science

in the subject of:
Electrical Engineering

Thesis Advisor:
Dr. Félix Francisco Ramos Corchado

Guadalajara, Jalisco, May 2008.

Animación de Criaturas Virtuales Usando Aprendizaje

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:
Moisés Uc Cetina
Licenciado en Ciencias de la Computación
Universidad Autónoma de Yucatán 1998-2003

Becario de CONACYT, expediente no. 187509

Director de Tesis:
Dr. Félix Francisco Ramos Corchado

CINVESTAV del IPN Unidad Guadalajara, Mayo de 2008.

Animation of Virtual Creatures Using Learning

Master of Science Thesis
In Electrical Engineering

By:
Moisés Uc Cetina
B.Sc. in Computer Science
Universidad Autónoma de Yucatán 1998-2003

Scholarship granted by CONACYT, No. 187509

Thesis Advisor:
Dr. Félix Francisco Ramos Corchado

CINVESTAV del IPN Unidad Guadalajara, May, 2008.

Resumen

En esta tesis se presenta una metodología novedosa para la *Animación de Criaturas Virtuales Usando Aprendizaje*. Esta metodología se basa en el uso de un algoritmo de *Aprendizaje por Refuerzo* muy conocido, tal como lo es *Q-learning* (Watkins, 1989), para animar criaturas virtuales articuladas.

Se presentan dos casos de estudio en los que se aplicó exitosamente esta metodología novedosa, obteniendo resultados alentadores.

Se utiliza esta experiencia, así como cierto conocimiento previo tal como *ecuaciones de geometría plana*, para proponer otra metodología novedosa que provea a nuestra criatura virtual la capacidad de *aprender a dibujar una figura geométrica en un plano* $z = c$, simulando de esta forma a una persona que está de pie frente a una pizarra dibujando una figura geométrica en ella. Quizás un maestro de matemáticas impartiendo una clase de geometría plana.

Se aplica esta nueva metodología a la criatura virtual presentada en nuestro segundo caso de estudio: *El Brazo Humano con Cuatro Grados de Libertad*, generando el siguiente *Alfabeto de Movimiento*:

$\Sigma = \{$ `circle, semicircle, quartercircle, hline, vline, sline, bsline` $\}$

Este alfabeto es muy importante porque se puede utilizar como entrada de otro algoritmo de aprendizaje o de planeación (que trabaja en un nivel más alto) para obtener animaciones más complejas de dicha criatura virtual, tales como escribir en una pizarra o en una hoja de papel.

Abstract

In this thesis we present a novel methodology for the *Animation of Virtual Creatures Using Learning*. Our methodology is based on the use of a well-known *Reinforcement Learning* algorithm such as *Q-learning* (Watkins, 1989), to animate articulated virtual creatures.

We present two case studies in which we have successfully applied our novel methodology, obtaining encouraging results.

We use this experience, as well as some previous knowledge like *plane geometry equations*, to propose another novel methodology that provides our virtual creature with the capability of *learning to draw geometric figures in a plane* $z = c$, simulating in this way a person who is standing in front of a blackboard drawing a geometric figure on it. Perhaps a math teacher giving a plane geometry's lecture.

We apply this new methodology to the virtual creature presented in our second case study: *The Human Arm with Four Degrees of Freedom*, generating the following *Alphabet of Movement*:

$\Sigma = \{$ `circle, semicircle, quartercircle, hline, vline, sline, bsline` $\}$

This alphabet is very important because it can be used as input to another learning or planning algorithm (working in a higher level) to obtain more complex animations of such virtual creature, like writing on a blackboard or on a sheet of paper.

Acknowledgments

First of all, I would like to thank **God** for giving me the opportunity to finish this thesis.

I would especially like to thank my family for being the greatest support in my life. In particular, I would like to thank:

my father, **Víctor**, for being my source of inspiration,

my mother, **Wilma**, for being my strength,

my sister, **Flor**, for teaching me to never give up,

my brother, **Barón**, for showing me the road to success,

my brother, **Rodrigo**, for teaching me that the hope is the last to die,

my nephew, **Alexis**, for his good desires in very difficult moments, and

my niece, **Sofía**, for recieving me in her home with open arms.

I would also like to thank my girlfriend, **Liz**, for her love and unconditional support.

I must thank my advisor, **Félix Ramos**, for all his support and the time he inverted in me.

Thanks to **CONACYT** for his financial support in the form of scholarship number 187509.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter we aim to introduce the reader to the problem: *Animation of Virtual Creatures Using Learning.* First we present the problem description, then we present the thesis objective and finally, in the last section, we present the thesis structure.

1.1 Problem Description

One of the great challenges of our time is to create virtual characters that can reason, feel, learn and react as if they were real living creatures. The design of these virtual characters has been a motivating task for researchers of different areas like *Robotics*, *Virtual Reality*, *Artificial Intelligence*, *Computer Graphics* and *Computer Animation.*

The advances in such research areas have been impressive but there is still a lot of work to be done. In *the video game industry* the users demand everyday more sophisticated video games in which they can enhance their presence in the virtual environment in which they navigate, perceive elements and interact with the virtual characters. In *the film industry* there exists a growing interest to characterize actors in animated movies in a more realistic way.

In order to attain this, we propose a novel approach to the animation of virtual characters or creatures. Our approach is based on the use of *Machine Learning* algorithms to provide the virtual creature with the capability of learning a new skill.

1.2 Thesis Objective

The **main objective** of this thesis is:

- *To propose a methodology for the animation of virtual creatures using a Machine Learning algorithm.*

In addition to the main objective, we have some other **particular objectives**:

- To select a *Machine Learning* algorithm that fits well the problem of animating virtual creatures.
- To define at least one case study for the *Machine Learning* algorithm selected.
- To use the *Machine Learning* algorithm to provide the virtual creature of our case study with the capability of learning a new skill.
- To generate an *Alphabet of Movement* for the virtual creature of our case study.

1.3 Thesis Structure

This thesis is structured as follows:

- **Chapter 2: State of the Art.** Presents a revision of the literature on machine learning approaches, as well as some previous works that apply machine learning methods to virtual creatures.
- **Chapter 3: Proposed Methodology.** Presents our proposed methodology for solving the animation of virtual creatures using learning.
- **Chapter 4: The Human Arm with Two Degrees of Freedom.** Presents our first case study.
- **Chapter 5: The Human Arm with Four Degrees of Freedom.** Presents our second case study.
- **Chapter 6: Towards an Alphabet of Movement.** Presents a methodology for generating an alphabet of movement for the virtual creature of our second case study.
- **Chapter 7: Conclusions and Future Work.** Presents the thesis conclusions and the future work.

Chapter 2

State of the Art

In this chapter we present the state of the art in: *Machine Learning methods applied to the Animation of Virtual Creatures.* First we present some Machine Learning approaches, then we present some works that apply Machine Learning methods to Virtual Creatures and finally, in the last section, we present the conclusions.

2.1 Introduction

The problem of: *Animation of Virtual Creatures Using Learning*, involves many research areas like Robotics, Virtual Reality, Artificial Intelligence, Computer Graphics and Computer Animation among others. Therefore, in order to present a state of the art for this problem, we proceed in the following manner: first we present some *Machine Learning* approaches (section 2.2) and then we present some works in which *Machine Learning* algorithms are applied to virtual creatures (section 2.3).

2.2 Machine Learning Approaches

Machine Learning is the study of computer algorithms that improve automatically through experience. Applications range from datamining programs that discover general rules in large data sets, to information filtering systems that automatically learn users' interests [12]. In the following subsections 2.2.1 to 2.2.6 we present some Machine Learning approaches.

2.2.1 Genetic Algorithms

Genetic algorithms (GAS) provide a learning method motivated by an analogy to biological evolution. Rather than search from general-to-specific hypotheses, or from simple-to-complex, GAS generate successor hypotheses by repeatedly mutating and recombining parts of the best currently known hypotheses. At each step, a collection of hypotheses called the current *population* is updated by replacing some fraction of the population by offspring of the most fit current hypotheses. The process forms a generate-and-test beam-search of hypotheses, in which variants of the best current hypotheses are most likely to be considered next [12].

The popularity of GAS is motivated by a number of factors including:

- Evolution is known to be a successful, robust method for adaptation within biological systems.
- GAS can search spaces of hypotheses containing complex interacting parts, where the impact of each part on overall hypothesis fitness may be difficult to model.
- Genetic algorithms are easily parallelized and can take advantage of the decreasing costs of powerful computer hardware.

2.2.2 Learnable Evolution Model

The *Learnable Evolution Model* [11] or *LEM* is a new class of evolutionary computation processes. In contrast to Darwinian-type evolution that relies on mutation, recombination, and selection operators, LEM employs machine learning to generate new populations. Specifically, in *Machine Learning mode*, a learning system seeks reasons why certain individuals in a population (or a collection of past populations) are superior to others in performing a designated class of tasks. These reasons, expressed as inductive hypotheses, are used to generate new populations.

A remarkable property of LEM is that it is capable of quantum leaps ("insight jumps") of the fitness function, unlike Darwinian-type evolution that typically proceeds through numerous slight improvements. In our early experimental studies, LEM significantly outperformed evolutionary computation methods used in the experiments, sometimes achieving speed-ups of two or more orders of magnitude in terms of the number of evolutionary steps [11].

LEM has a potential for a wide range of applications, in particular, in such domains as complex optimization or search problems, engineering design, drug design, evolvable hardware, software engineering, economics, data mining, and automatic programming [11].

2.2.3 Learning Classifier Systems

Learning classifier systems are a Machine Learning *paradigm* introduced by John Holland in 1978. In learning classifier systems an agent learns to perform a certain task by *interacting* with a partially unknown environment from which the agent receives feedback in the form of numerical *reward*. The incoming reward is exploited to guide the *evolution* of the agent's *behavior* which, in learning classifier systems, is represented by a set of rules, the *classifiers*. In particular, *temporal difference learning* is used to estimate the goodness of classifiers in terms of future reward; *genetic algorithms* are used to favour the reproduction and recombination of better classifiers [8].

2.2.4 Collaborative Logic-based Learning

Collaborative logic-based learning is a method for implementing *adaptability* (a fundamental property of any intelligent system) in multi-agent systems. This method is based on two building blocks: (1) a set of operations centred around inductive logic programming for generalizing agents' observations into sets of rules, and (2) a set of communication strategies for sharing acquired knowledge among agents in order to improve the collaborative learning process. Using these modular building blocks, several learning algorithms can be constructed with different trade-offs between the quality of learning, computation and communication requirements, and the disclosure of agent's private information. The method has been implemented as a modular software component that can be integrated into the control loop of an intelligent agent. The method has been evaluated on a simulated logistic scenario, in which teams of trading agents learn the properties of the environment in order to optimize their operation [9].

2.2.5 Incremental Learning

Incremental learning is a novel approach to the motion-planning problem that can learn incrementally on every planning query and effectively manage the learned roadmap as the process goes on. This planner is based on previous work on probabilistic roadmaps and uses a data structure called Reconfigurable Random Forest (RRF), which extends the Rapidly-exploring Random Tree (RRT) structure proposed in the literature. The planner can account for environmental changes while keeping the size of the roadmap small. The planner removes invalid nodes in the roadmap as the obstacle configurations change. It also uses a tree-pruning algorithm to trim RRF into a more concise representation. Our experiments show that the planner is flexible and efficient [10].

2.2.6 Reinforcement Learning

Reinforcement learning addresses the question of how an autonomous agent that senses and acts in its environment can learn to choose optimal actions to achieve its goals. This very generic problem covers tasks such as learning to control a mobile robot, learning to optimize operations in factories, and learning to play board games. Each time the agent performs an action in its environment, a trainer may provide a reward or penalty to indicate the desirability of the resulting state. For example, when training an agent to play a game the trainer might provide a positive reward when the game is won, negative reward when it is lost, and zero reward in all other states. The task of the agent is to learn from this indirect, delayed reward, to choose sequences of actions that produce the greatest cumulative reward [12].

Reinforcement learning is a computational approach to understanding and automating goal-directed learning and decision-making. It is distinguished from other computational approaches by its emphasis on *learning by the individual from direct interaction with its environment*, without relying on exemplary supervision or complete models of the environment. In our opinion, reinforcement learning is the first field to seriously address the computational issues that arise when learning from interaction with an environment in order to achieve long-term goals [16].

Reinforcement learning uses a formal framework defining the interaction between a learning agent and its environment in terms of *states*, *actions*, and *rewards*. This framework is intended to be a simple way of representing essential features of the artificial intelligence problem. These features include a sense of *cause and effect*, a sense of *uncertainty* and *nondeterminism*, and the existence of *explicit goals* [16].

The concepts of *value* and *value functions* are the key features of the reinforcement learning methods. We take the position that value functions are essential for efficient search in the space of policies. Their use of value functions distinguishes reinforcement learning methods from evolutionary methods that search directly in policy space guided by scalar evaluations of entire policies [16].

2.3 Machine Learning Approaches to Virtual Creatures

Here we present some works that apply Machine Learning methods to Virtual Creatures.

In [7], Reinforcement Learning Algorithms are applied for the generation of Autonomous Intelligent Virtual Robots, that can learn and enhance their task performance in assisting

humans in housekeeping. For the control system architecture of the virtual agents, two algorithms, based on Watkins' Q(λ) Learning and the Zeroth-Level Classifier System (ZLCS), are incorporated with Fuzzy Inference Systems(FIS). Performance of these algorithms is evaluated and compared. A 3D application of a Virtual Robot whose task is to interact with Virtual Humans and offer optimal services on everyday in-house needs is designed and implemented. The learning systems are incorporated in the decision-making process of the Virtual Robot Servant to allow itself to understand and evaluate the fuzzy value requirements and enhance its performance.

In [14], the potential for instructing animated agents through collaborative dialog in a simulated environment is described. The abilities to share activity with a human instructor and employ both verbal and nonverbal modes of communication allow the agent to be taught in a manner natural for the instructor. A system that uses such shared activity and instruction to acquire the knowledge needed by an agent is described. This system is implemented in STEVE, an embodied agent that teaches physical tasks to human students. STEVE begins learning about a task through a process of *programming by demonstration* (Cypher 1993). The human instructor tells STEVE to observe his actions, and then performs the task by manipulating objects in the simulated world. As the agent watches, it learns both necessary information about the environment and the procedural knowledge associated with the task.

In [3], two well-known Reinforcement Learning algorithms (*Q-Learning and TD-Learning*) are applied to a virtual environment as a behavioral engine for exploration, learning and visiting a virtual environment. Therefore, contrary to the use of reinforcement learning algorithms, our interest here lies more in learning than in the exploitation of this learning. Thus, it is indeed the learning rather than the optimization that allows us to simulate the behavior of Autonomous Virtual Agents (AVAs) and this constitutes a new use of reinforcement learning.

In [4], an approach based on the multi-sensory integration of the standard theory of neuroscience, where signals of a single object coming from distinct sensory systems are combined, is presented. The acquisition steps of signals, filtering, selection and simplification intervening before proprioception, active and predictive perception are integrated into virtual sensors and a virtual environment. This research is focused on two aspects: 1) the assignment problem: determining which sensory stimuli belong to the same virtual object and (2) the sensory recoding problem: recoding signals in a common format before combining them. Three novel methodologies to map the information coming from the virtual sensors of vision, audition and touch as well as that of the virtual environment in the form of a 'cognitive map' were developed.

In [5], a new integration approach to simulate an Autonomous Virtual Agent's cogni-

tive learning of a task for interactive Virtual Environment applications is proposed. This research is focused on the behavioural animation of virtual humans capable of acting independently. Its contribution is important because it presents a solution for fast learning with evolution. The concept of a Learning Unit Architecture that functions as a control unit of the Autonomous Virtual Agent's brain is proposed. Although this technique has proved to be effective in a case study, there is no guarantee that it will work for every imaginable Autonomous Virtual Agent and Virtual Environment. The results are illustrated in a domain that requires effective coordination of behaviours, such as driving a car inside a virtual city.

In [6], a new integration approach for simulation and behaviour in the learning context that is able to coherently manage the shared virtual environment for the simulation of autonomous virtual agents is proposed. This low-level learning technique has proved fast, simple and robust. It is also able to automatically learn behavioural models for difficult tasks. Thus, it is believed that it will be more useful to the computer graphics community than a technique based on the classical Q-learning approach. The results are illustrated in two case studies that require effective coordination of behaviours.

In [17], the architecture of the ANIMUS framework is described. This framework facilitates the creation of synthetic characters that convey the illusion of being *alive*. The components of ANIMUS are inspired by observations made in biological organisms, and provide means for creating autonomous agents that mimic awareness of their environment, of other agents, and of human audience. They also show particular roles, personality, and emotions, active and reactive behavior, automatic reflexes, and selective attention; use temporal memory and learning capabilities to evolve in their dynamic virtual worlds, and express their thought and emotions with a flexible animation system while they interact with the user in immersive 3D environments. ANIMUS creatures follow artistic conceptual designs and constraints that determine the way they behave, react and interact with other creatures and the user, allowing the designer to create meaningful and interesting characters. The framework can be applied to complex immersive environments like CAVE systems or other interactive applications like video games and advanced man-machine interfaces, providing high level tools for creating a new generation of responsive believable agents.

In [13], different approaches of reinforcement learning in terms of their applicability in humanoid robotics are discussed. Methods can be coarsely classified into three different categories, i.e., greedy methods, 'vanilla' policy gradient methods, and natural gradient methods. The greedy methods are not likely to scale into the domain humanoid robotics as they are problematic when used with function approximation. 'Vanilla' policy gradient methods on the other hand have been successfully applied on real-world robots including at least one humanoid robot. These methods can be significantly improved using the natural policy gradient instead of the regular policy gradient. A derivation of the natural policy gradient

is provided, proving that the average policy gradient of Kakade is indeed the true natural gradient. A general algorithm for estimating the natural gradient, the Natural Actor-Critic algorithm, is introduced. This algorithm converges to the nearest local minimum of the cost function with respect to the Fisher information metric under suitable conditions. The algorithm outperforms non-natural policy gradients by far in a cart-pole balancing evaluation, and for learning nonlinear dynamic motor primitives for humanoid robot control. It offers a promising route for the development of reinforcement learning for truly high-dimensionally continuous state-action systems.

In [18], reinforcement learning methods are applied to learn domain-specific heuristics for job shop scheduling. A repair-based scheduler starts with a critical-path schedule and incrementally repairs constraint violations with the goal of finding a short conflict-free schedule. The temporal difference algorithm TD(λ) is applied to train a neural network to learn a heuristic evaluation function over states. This evaluation function is used by a one-step lookahead search procedure to find good solutions to new scheduling problems. This approach is evaluated on synthetic problems and on problems from a NASA space shuttle payload processing task. The evaluation function is trained on problems involving a small number of jobs and then tested on larger problems. The TD scheduler performs better than the best known existing algorithm for this task-Zweben's iterative repair method based on simulated annealing. The results suggest that reinforcement learning can provide a new method for constructing high-performance scheduling systems.

2.4 Conclusions

After reviewing the literature, we have decided to use a *Reinforcement Learning* algorithm, in particular we have decided to use the well-known *Q-learning* algorithm (Watkins, 1989) as a key algorithm to approach a solution for solving the objective presented in this thesis.

In the following chapter we present in detail our proposed methodology for solving the problem: *Animation of Virtual Creatures using Learning.*

Chapter 3

Proposed Methodology

In this chapter we present our proposed methodology for solving the problem: *Animation of Virtual Creatures Using Learning*. This methodology consists in five steps as we can see in Figure 3.1. Each one of these steps is explained in detail in sections 3.1 to 3.5.

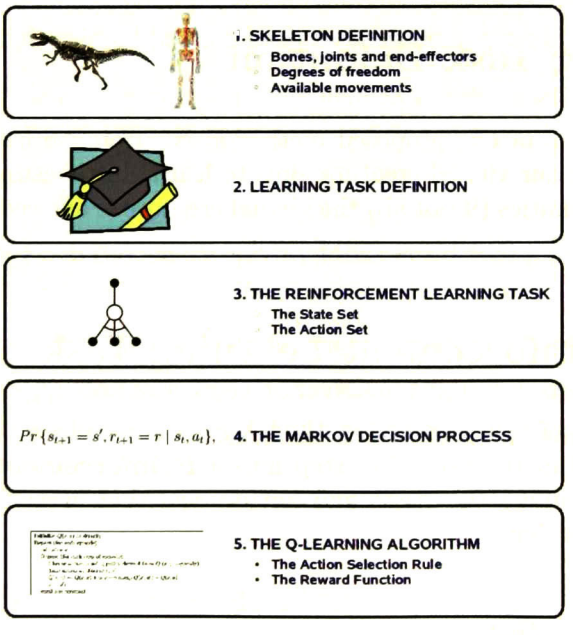


Figure 3.1: The five-step methodology.

3.1 Skeleton Definition

This is the first step of our proposed methodology. Here we have to completely define the skeleton's structure of the virtual creature that we aim to animate. By *completely define* we mean:

- defining the number of **bones** and **joints**, their names and relations,
- defining the number of **end-effectors** and their locations in the skeleton,
- defining the number of **degrees of freedom** assigned to each joint, and finally
- defining the set of **available movements** for the virtual creature depending on the number of degrees of freedom previously defined.

After having completely defined the structure of the skeleton, we have to implement such structure with **Java 3D** [2].

3.2 Learning Task Definition

This is the second step of our proposed methodology. Here we have to define as clear as possible the task that our virtual creature aims to learn. In this step is also very important to specify which extremities (if not all) the virtual creature is allowed to use in order to learn such task.

3.3 The Reinforcement Learning Task

This is the third step of our proposed methodology. Here we have to translate the learning task formally defined in the preceding step into a **Reinforcement Learning Task**. By *translate* we mean defining the **state** and **action** sets that every Reinforcement Learning Task well defined must have.

3.3.1 The State Set

Here we represent a state s as a n-tuple ($\texttt{joMovementAng}_0$, $\texttt{joMovementAng}_1$, $\ldots$, $\texttt{joMovementAng}_{n-1}$), where:

- $\texttt{joMovementAng}_i \in \{0°, 10°, 20°, \ldots, UpperBound_i\}$ is the angle assigned to the movement `Movement` of the joint `jo`, and
- n is the total number of degrees of freedom of the whole skeleton.

Therefore, the **state set** is

- $\mathcal{S} = \{0°, 10°, 20°, \ldots, UpperBound_0\} \times \{0°, 10°, 20°, \ldots, UpperBound_1\} \times \ldots \times \{0°, 10°, 20°, \ldots, UpperBound_{n-1}\}$, and
- $|\mathcal{S}| = |\{0°, 10°, 20°, \ldots, UpperBound_0\}| \bullet |\{0°, 10°, 20°, \ldots, UpperBound_1\}| \bullet \ldots \bullet |\{0°, 10°, 20°, \ldots, UpperBound_{n-1}\}|$.

3.3.2 The Action Set

Here we have to translate *the set of available movements* into *the set of actions* of the Reinforcement Learning Task. Thus, each movement is translated into two actions:

1. to *increase* the angle related to such movement in ten degrees.
2. to *decrease* the angle related to such movement in ten degrees.

Therefore, the **action set** is

- $\mathcal{A} = \{\texttt{joMovement+}_0, \texttt{joMovement-}_1, \ldots, \texttt{joMovement+}_{2n-2}, \texttt{joMovement-}_{2n-1}\}$, where:
  - $\texttt{joMovement+}_i$: increase angle `joMovementAng` in ten degrees.
  - $\texttt{joMovement-}_{i+1}$: decrease angle `joMovementAng` in ten degrees.
  - n: total number of degrees of freedom of the whole skeleton.
- And, $|\mathcal{A}| = 2n$.

3.4 The Markov Decision Process

This is the fourth step of our proposed methodology. Here we have to verify if the Reinforcement Learning Task defined in step 3 (section 3.3) is a *finite Markov decision process (finite MDP)*.

Finite MDPs are particularly important to the theory of reinforcement learning; they are all you need to understand 90% of modern reinforcement learning [16].

First we have to check if the state signal defined in step 3 (section 3.3.1) has the *Markov property.* In that case, the whole Reinforcement Learning Task is also said to have the Markov property and therefore it is called a *Markov decision process*, or *MDP* Furthermore, if its state and action sets defined in step 3 (section 3.3) are finite, then it is called a *finite Markov decision process (finite MDP).*

3.5 The Q-learning Algorithm

This is the fifth and last step of our proposed methodology. Here we have to solve the Reinforcement Learning Task defined in step 3 (section 3.3) using a Reinforcement Learning Algorithm.

We propose the use of a well-known Reinforcement Learning Algorithm such as *Q-learning* (Watkins, 1989). Its simplest form, *one-step Q-learning*, is defined by

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

We propose to set the values of α and γ to 0.5.

In this algorithm, the learned action-value function, Q, directly approximates Q^*, the optimal action-value function, independent of the policy being followed. The policy still has an effect in that it determines which state-action pairs are visited and updated. However, all that is required for correct convergence is that all pairs continue to be updated. This is a minimal requirement in the sense that any method guaranteed to find optimal behavior in the general case must require it. Under this assumption and a variant of the usual stochastic approximation conditions on the sequence of step-size parameters, Q_t has been shown to converge with probability 1 to Q^* The Q-learning algorithm is shown in procedural form in Figure 3.2 [16].

We propose to implement the Q-learning algorithm with **Java** [1].

3.5.1 The Action Selection Rule

Here we have to define an action selection rule for the Q-learning algorithm. We propose to balance *exploration* and *exploitation* by choosing an ϵ-*greedy* method. However, instead of

Initialize $Q(s,a)$ **arbitrarily**
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a **from** s **using policy derived from** Q **(e.g.,** ε**-greedy)**
    Take action a**, observe** r, s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
    $s \leftarrow s'$;
  until s **is terminal**

Figure 3.2: The Q-learning algorithm in procedural form.

assigning ϵ a constant value during the whole algorithm's execution (as in most cases), in this particular case we propose to reduce ϵ over time to try to get the best of both high and low values. In other words, we propose to initialize ϵ with the real value of 1.0 and at the beginning of each episode to decrease it in the following manner:

$$\epsilon = 1.0 - \left(\frac{elapsed\ episodes}{total\ episodes}\right)$$

Therefore, we propose to update the value of ϵ in a way inversely proportional to the number of elapsed episodes in the algorithm's execution.

3.5.2 The Reward Function

Here we have to define a reward function for the Q-learning algorithm. The reward function is one of the most important components of every Reinforcement Learning Algorithm because it defines the purpose or goal of the agent in a Reinforcement Learning Task.

Therefore, we need to define a reward function that closely represents the goal of the agent in our Reinforcement Learning Task.

3.6 Conclusions

In this chapter we have presented our five-step methodology for solving the problem: *Animation of Virtual Creatures Using Learning.* Notice that step five (*The Q-learning Algorithm*), is a key step in our methodology because there, we have to define a *reward function* that

closely represents the *learning task* defined in step two. Therefore, our results depend on how well such *reward function* was defined.

In the following two chapters, we present two case studies in which we applied this methodology successfully and we obtained encouraging results.

Chapter 4

The Human Arm with Two Degrees of Freedom

In this chapter we present our first case study: *The Human Arm with Two Degrees of Freedom.* We start by giving a description of the problem and a definition of the task to solve, then we give a representation of the task as a *Reinforcement Learning Task.* Once we have this representation, we proceed to the implementation of the task with a *Q-learning* algorithm and finally, in the last section, we show the simulation of this task and the obtained results.

4.1 Problem Description

The case study proposed in this chapter consists in a virtual creature simulating a human baby who is learning to move his superior extremities in a coordinated way. This with the objective of satisfying a basic physiological necessity such as eating a chicken soup in a proper manner using a spoon.

At first sight this problem may appear to be simple, however after a deep analysis we realized that it could be subdivided into at least five separate complex tasks:

1. Learning to move the arm to reach the spoon.
2. Learning to grasp the spoon.
3. Learning to carry the spoon into the soup bowl.
4. Learning to take a little of soup with the spoon.
5. Learning to carry the spoon to the mouth without dropping the soup.

As our first approximation to this problem, we decided to solve the first complex task, the one which aims *learning to move the arm to reach the spoon.*

In order to solve this complex task, we used the methodology presented in chapter 3. The results of applying such methodology to this complex task are presented in sections 4.2 to 4.6.

4.2 Skeleton Definition

For simplicity, we focused only on the virtual creature's left arm, which is composed of two bones: **humerus** and **forearm**, and two joints: **shoulder** and **elbow**. The arm's **end-effector** is considered to be the forearm's end. Figure 4.1 shows the implementation of this virtual creature in Java 3D [2].

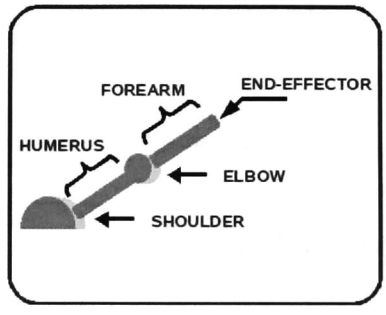


Figure 4.1: Skeleton's structure for The Human Arm with Two DOF

The available movements for this arm depend on the number of degrees of freedom assigned to each joint. As this is our first case study, we considered **two degrees of freedom**, one of them assigned to the shoulder and the other one assigned to the elbow. Therefore, the **available movements** for this arm are:

- For the **shoulder**:
  - Pitch (up or down).
- For the **elbow**:
  - Pitch (up or down).

Figure 4.2 shows the available movements for the arm in Java 3D [2].

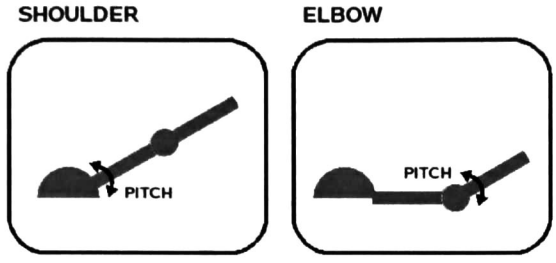


Figure 4.2: Available movements for The Human Arm with Two DOF

4.3 Learning Task Definition

The task consists in learning to move the virtual creature's left arm from an initial position $\mathbf{I_{x,y}}$ (end-effector's initial position) to a goal position $\mathbf{G_{x,y}}$ (end-effector's goal position). Figure 4.3 shows a representation of this learning task.

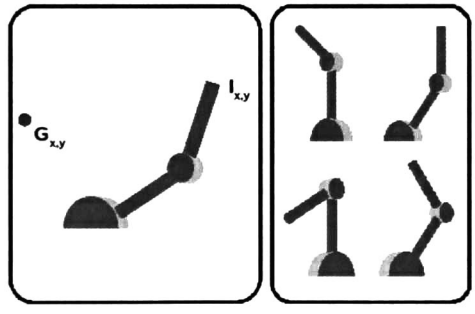


Figure 4.3: Learning task for The Human Arm with Two DOF.

4.4 The Reinforcement Learning Task

In the preceding section we formally defined our learning task. Now we need to translate this learning task into a **Reinforcement Learning Task**.

4.4.1 The State Set

As we stated before, for this learning task we have considered two degrees of freedom. Therefore, it is useful to represent a state s as a 2-tuple (`shPitchAng, elPitchAng`), where:

- `shPitchAng` $\in \{0°, 10°, 20°, \ldots, 180°\}$ is the shoulder's pitch angle.
- `elPitchAng` $\in \{0°, 10°, 20°, \ldots, 180°\}$ is the elbow's pitch angle.

Therefore, the **state set** is

- $\mathcal{S} = \{0°, 10°, 20°, \ldots, 180°\} \times \{0°, 10°, 20°, \ldots, 180°\}$, and
- $|\mathcal{S}| = 19 \times 19 = 361$.

4.4.2 The Action Set

The action set must represent all the available movements of the arm considered in this learning task.

Therefore, the **action set** is

- $\mathcal{A} =$ { `shPitchUp, shPitchDown, elPitchUp, elPitchDown` }, where:
  - `shPitchUp`: pitch up the shoulder ten degrees.
  - `shPitchDown`: pitch down the shoulder ten degrees.
  - `elPitchUp`: pitch up the elbow ten degrees.
  - `elPitchDown`: pitch down the elbow ten degrees.
- And, $|\mathcal{A}| = 4$.

4.5 The Markov Decision Process

We assume here that the state signal defined in section 4.4.1 has the *Markov property*. Therefore, the whole Reinforcement Learning Task defined in section 4.4 is also said to have the Markov property.

Now, as this Reinforcement Learning Task satisfies the Markov property, then it is called a *Markov decision process*, or *MDP*. Furthermore, as its state and action sets defined in section 4.4 are finite, then it is called a *finite Markov decision process (finite MDP)*.

In order to completely define this finite MDP, we only need to define the one-step dynamics of its environment. Given any state and action, s and a, the probability of each possible next state, s', is

$$\mathcal{P}^a_{ss'} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}$$

Similarly, given any current state and action, s and a, together with any next state, s', the expected value of the next reward is

$$\mathcal{R}^a_{ss'} = E\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\}$$

4.6 The Q-learning Algorithm

In order to solve this Reinforcement Learning Task, we chose a well-known Reinforcement Learning Algorithm such as *one-step Q-learning*, defined by the following action-value function:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

The values of α and γ were set to 0.5.

4.6.1 The Action Selection Rule

In order to balance *exploration* and *exploitation* we chose an ϵ-*greedy* method. However, instead of assigning ϵ a constant value during the whole algorithm's execution (as in most cases), in this particular case we reduce ϵ over time to try to get the best of both high and low values. In other words, ϵ is initialized with the real value of 1.0 and at the beginning of each episode it is decreased in the following manner:

$$\epsilon = 1.0 - \left(\frac{\textit{elapsed episodes}}{\textit{total episodes}} \right)$$

Therefore, the value of ϵ is updated in a way inversely proportional to the number of elapsed episodes in the algorithm's execution.

4.6.2 The Reward Function

The reward function is one of the most important components of every Reinforcement Learning Algorithm because it defines the purpose or goal of the agent in a Reinforcement Learning Task.

Therefore, we need to define a reward function that closely represents the goal of the agent in our Reinforcement Learning Task.

As our first approximation, we defined our *reward function* $\mathcal{R}$ as the **Euclidean distance** between $\mathbf{C_{x,y}}$ (end-effector's current position) and $\mathbf{G_{x,y}}$ (end-effector's goal position). Thus, the goal of our Reinforcement Learning Task is to minimize the Euclidean distance between $C_{x,y}$ and $G_{x,y}$. In other words, **to minimize** the total reward received in the long run.

Although this is a good approximation, there is a little problem, the goal of every Reinforcement Learning Algorithm is **to maximize** the total reward received in the long run which is exactly the opposite of the goal of our Reinforcement Learning Task.

In order to fix this problem, we simply used the **Euclidean distance** as a **negative reward**. Therefore, we finally defined our *reward function* $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}^-$ as:

$$\mathcal{R}(s, a, s') = -\sqrt{(\mathbf{c}_x - \mathbf{g}_x)^2 + (\mathbf{c}_y - \mathbf{g}_y)^2}$$

So far, we have correctly defined our reward function. However, $\mathbf{C_{x,y}}$ is unknown and needs to be calculated from s' That is, at each time step, we need to calculate the end-effector's current position $\mathbf{C_{x,y}}$, given the arm's current state s' This problem is known as **The Forward Kinematics Problem** [15] and will be discussed in the following section.

4.6.3 The Forward Kinematics Problem

In order to solve this Forward Kinematics Problem, we used the **Denavit-Hartenberg convention (D-H convention)** [15] to select the frames attached to each link in a systematic way.

We first establish the joint coordinate frames using the D-H convention as shown in Figure 4.4. The link parameters are shown in Table 4.1.

Table 4.1: DH parameters for The Human Arm with Two DOF.

Link	a_i	α_i	d_i	θ_i
1	a_1	0	0	θ_1
2	a_2	0	0	θ_2

The corresponding *A-matrices* are

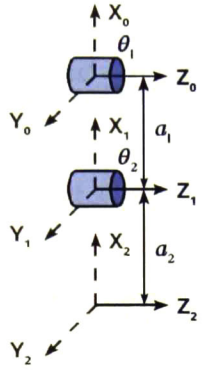


Figure 4.4: DH coordinate frame assignment for The Human Arm with Two DOF.

$$A_1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & a_1\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & 0 & a_1\sin\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & a_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & a_2\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The *T-matrices* are thus given by

$$T_1^0 = A_1$$

$$T_2^0 = A_1A_2 = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 & a_1\cos\theta_1 + a_2\cos(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 & a_1\sin\theta_1 + a_2\sin(\theta_1+\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Therefore the forward kinematics of the Human Arm with Two DOF is described by

$$T_2^0 = A_1 A_2 = \begin{bmatrix} r_{11} & r_{12} & 0 & \mathbf{c}_x \\ r_{21} & r_{22} & 0 & \mathbf{c}_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

$$\begin{aligned} \mathbf{c}_x &= a_1 \cos\theta_1 + a_2 \cos(\theta_1 + \theta_2) \\ \mathbf{c}_y &= a_1 \sin\theta_1 + a_2 \sin(\theta_1 + \theta_2). \end{aligned} \tag{4.1}$$

4.7 Conclusions

In this chapter we have successfully applied our five-step methodology (chapter 3) to our first case study: *The Human Arm with Two Degrees of Freedom.* The kind of movements we obtained from this case study are *movements in 2D.* Although the animation of this virtual creature was in a 3D environment, the kind of movements it performs are in a 2D plane, due to the number of degrees of freedom.

After running several experiments with this case study, we realized that our learning algorithm performed pretty well and the resulting animation of this virtual creature was very real. A key fact of our learning algorithm was defining the reward function correctly.

Although this case study may seem to be small, actually it was very helpful to our research work because it opened the path for future research.

Once we have successfully applied our methodology to this case study, our new goal is to propose a new case study which represents a greater challenge for our methodology and where we can obtain movements much more complex. In order to attain this, we should increase the complexity of our virtual creature, that is, we should increase its number of degrees of freedom.

In the following chapter, we present our second case study: *The Human Arm with Four Degrees of Freedom*, we apply our five-step methodology to such case study.

Chapter 5

The Human Arm with Four Degrees of Freedom

In this chapter we present our second case study: *The Human Arm with Four Degrees of Freedom*. We start by giving a description of the problem and a definition of the task to solve, then we give a representation of the task as a *Reinforcement Learning Task*. Once we have this representation, we proceed to the implementation of the task with a *Q-learning* algorithm and finally, in the last section, we show the simulation of this task and the obtained results.

5.1 Problem Description

The case study proposed in this chapter consists in a virtual creature simulating a human baby who is learning to move his superior extremities in a coordinated way. This with the objective of satisfying a basic physiological necessity such as eating a chicken soup in a proper manner using a spoon.

At first sight this problem may appear to be simple, however after a deep analysis we realized that it could be subdivided into at least five separate complex tasks:

1. Learning to move the arm to reach the spoon.
2. Learning to grasp the spoon.
3. Learning to carry the spoon into the soup bowl.
4. Learning to take a little of soup with the spoon.

5. Learning to carry the spoon to the mouth without dropping the soup.

As our first approximation to this problem, we decided to solve the first complex task, the one which aims *learning to move the arm to reach the spoon.*

In order to solve this complex task, we used the methodology presented in chapter 3. The results of applying such methodology to this complex task are presented in sections 5.2 to 5.6.

5.2 Skeleton Definition

For simplicity, we focused only on the virtual creature's left arm, which is composed of two bones: **humerus** and **forearm**, and two joints: **shoulder** and **elbow**. The arm's **end-effector** is considered to be the forearm's end. Figure 5.1 shows the implementation of this virtual creature in Java 3D [2].

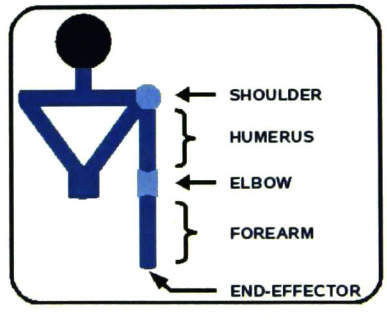


Figure 5.1: Skeleton's structure for The Human Arm with Four DOF.

The available movements for this arm depend on the number of degrees of freedom assigned to each joint. As this is our second case study, we considered **four degrees of freedom**, three of them assigned to the shoulder and the other one assigned to the elbow. Therefore, the **available movements** for this arm are:

- For the **shoulder**:
  - Roll (left or right).
  - Yaw (left or right).
  - Pitch (up or down).

- For the **elbow**:
  - Pitch (up or down).

Figure 5.2 shows the available movements for the arm in Java 3D [2].

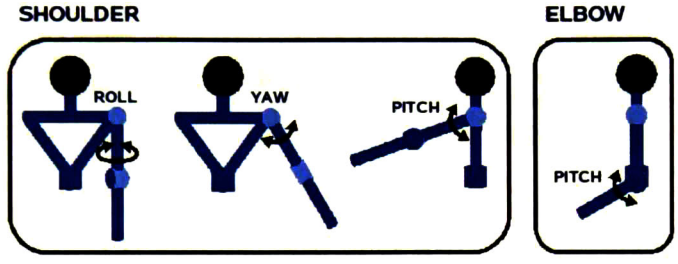


Figure 5.2: Available movements for The Human Arm with Four DOF.

5.3 Learning Task Definition

The task consists in learning to move the virtual creature's left arm from an initial position $\mathbf{I_{x,y,z}}$ (end-effector's initial position) to a goal position $\mathbf{G_{x,y,z}}$ (end-effector's goal position). Figure 5.3 shows a representation of this learning task.

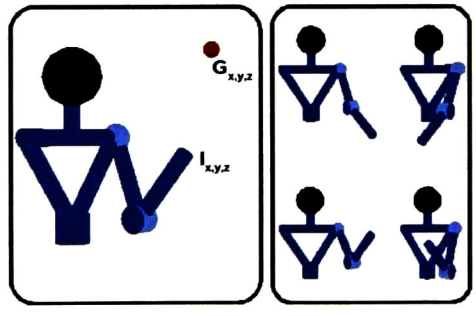


Figure 5.3: Learning task for The Human Arm with Four DOF.

5.4 The Reinforcement Learning Task

In the preceding section we formally defined our learning task. Now we need to translate this learning task into a **Reinforcement Learning Task**.

5.4.1 The State Set

As we stated before, for this learning task we have considered four degrees of freedom. Therefore, it is useful to represent a state s as a 4-tuple (`shRollAng`, `shYawAng`, `shPitchAng`, `elPitchAng`), where:

- `shRollAng` $\in \{0°, 10°, 20°, \ldots, 90°\}$ is the shoulder's roll angle.
- `shYawAng` $\in \{0°, 10°, 20°, \ldots, 180°\}$ is the shoulder's yaw angle.
- `shPitchAng` $\in \{0°, 10°, 20°, \ldots, 240°\}$ is the shoulder's pitch angle.
- `elPitchAng` $\in \{0°, 10°, 20°, \ldots, 150°\}$ is the elbow's pitch angle.

Therefore, the **state set** is

- $\mathcal{S} = \{0°, 10°, 20°, \ldots, 90°\} \times \{0°, 10°, 20°, \ldots, 180°\} \times \{0°, 10°, 20°, \ldots, 240°\} \times \{0°, 10°, 20°, \ldots, 150°\}$, and
- $|\mathcal{S}| = 10 \times 19 \times 25 \times 16 = 76000$.

5.4.2 The Action Set

The action set must represent all the available movements of the arm considered in this learning task.

Therefore, the **action set** is

- $\mathcal{A} =$ { `shRollLeft`, `shRollRight`, `shYawLeft`, `shYawRight`, `shPitchUp`, `shPitchDown`, `elPitchUp`, `elPitchDown` }, where:
  - `shRollLeft`: roll left the shoulder ten degrees.
  - `shRollRight`: roll right the shoulder ten degrees.
  - `shYawLeft`: yaw left the shoulder ten degrees.
  - `shYawRight`: yaw right the shoulder ten degrees.
  - `shPitchUp`: pitch up the shoulder ten degrees.
  - `shPitchDown`: pitch down the shoulder ten degrees.
  - `elPitchUp`: pitch up the elbow ten degrees.
  - `elPitchDown`: pitch down the elbow ten degrees.
- And, $|\mathcal{A}| = 8$.

5.5 The Markov Decision Process

We assume here that the state signal defined in section 5.4.1 has the *Markov property.* Therefore, the whole Reinforcement Learning Task defined in section 5.4 is also said to have the Markov property.

Now, as this Reinforcement Learning Task satisfies the Markov property, then it is called a *Markov decision process*, or *MDP.* Furthermore, as its state and action sets defined in section 5.4 are finite, then it is called a *finite Markov decision process (finite MDP).*

In order to completely define this finite MDP, we only need to define the one-step dynamics of its environment. Given any state and action, s and a, the probability of each possible next state, s', is

$$\mathcal{P}^a_{ss'} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}$$

Similarly, given any current state and action, s and a, together with any next state, s', the expected value of the next reward is

$$\mathcal{R}^a_{ss'} = E\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\}$$

5.6 The Q-learning Algorithm

In order to solve this Reinforcement Learning Task, we chose a well-known Reinforcement Learning Algorithm such as *one-step Q-learning*, defined by the following action-value function:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

The values of α and γ were set to 0.5.

5.6.1 The Action Selection Rule

In order to balance *exploration* and *exploitation* we chose an *ϵ-greedy* method. However, instead of assigning ϵ a constant value during the whole algorithm's execution (as in most cases), in this particular case we reduce ϵ over time to try to get the best of both high and

low values. In other words, ϵ is initialized with the real value of 1.0 and at the beginning of each episode it is decreased in the following manner:

$$\epsilon = 1.0 - \left(\frac{\textit{elapsed episodes}}{\textit{total episodes}}\right)$$

Therefore, the value of ϵ is updated in a way inversely proportional to the number of elapsed episodes in the algorithm's execution.

5.6.2 The Reward Function

The reward function is one of the most important components of every Reinforcement Learning Algorithm because it defines the purpose or goal of the agent in a Reinforcement Learning Task.

Therefore, we need to define a reward function that closely represents the goal of the agent in our Reinforcement Learning Task.

As our first approximation, we defined our *reward function* $\mathcal{R}$ as the **Euclidean distance** between $\mathbf{C_{x,y,z}}$ (end-effector's current position) and $\mathbf{G_{x,y,z}}$ (end-effector's goal position). Thus, the goal of our Reinforcement Learning Task is to minimize the Euclidean distance between $C_{x,y,z}$ and $G_{x,y,z}$. In other words, **to minimize** the total reward received in the long run.

Although this is a good approximation, there is a little problem, the goal of every Reinforcement Learning Algorithm is **to maximize** the total reward received in the long run which is exactly the opposite of the goal of our Reinforcement Learning Task.

In order to fix this problem, we simply used the **Euclidean distance** as a **negative reward**. Therefore, we finally defined our *reward function* $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}^-$ as:

$$\mathcal{R}(s, a, s') = -\sqrt{(\mathbf{c}_x - \mathbf{g}_x)^2 + (\mathbf{c}_y - \mathbf{g}_y)^2 + (\mathbf{c}_z - \mathbf{g}_z)^2}$$

So far, we have correctly defined our reward function. However, $\mathbf{C_{x,y,z}}$ is unknown and needs to be calculated from s'. That is, at each time step, we need to calculate the end-effector's current position $\mathbf{C_{x,y,z}}$, given the arm's current state s'. This problem is known as **The Forward Kinematics Problem** [15] and will be discussed in the following section.

5.6.3 The Forward Kinematics Problem

In order to solve this Forward Kinematics Problem, we used the **Denavit-Hartenberg convention (D-H convention)** [15] to select the frames attached to each link in a systematic way.

We first establish the joint coordinate frames using the D-H convention as shown in Figure 5.4. The link parameters are shown in Table 5.1.

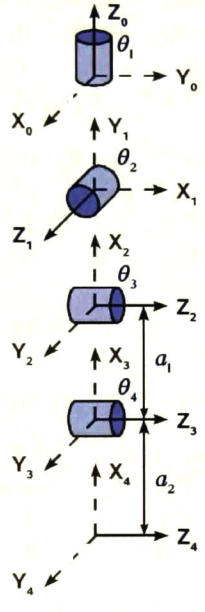


Figure 5.4: DH coordinate frame assignment for The Human Arm with Four DOF.

Table 5.1: DH parameters for The Human Arm with Four DOF.

Link	a_i	α_i	d_i	θ_i
1	0	90	0	θ_1
2	0	90	0	θ_2
3	a_1	0	0	θ_3
4	a_2	0	0	θ_4

The corresponding *A-matrices* are

$$A_1 = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 & 0 \\ \sin\theta_1 & 0 & -\cos\theta_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} \cos\theta_2 & 0 & \sin\theta_2 & 0 \\ \sin\theta_2 & 0 & -\cos\theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & a_1\cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & a_1\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & a_2\cos\theta_4 \\ \sin\theta_4 & \cos\theta_4 & 0 & a_2\sin\theta_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The *T-matrices* are thus given by

$$T_2^0 = A_1A_2 = \begin{bmatrix} \cos\theta_1\cos\theta_2 & \sin\theta_1 & \cos\theta_1\sin\theta_2 & 0 \\ \sin\theta_1\cos\theta_2 & -\cos\theta_1 & \sin\theta_1\sin\theta_2 & 0 \\ \sin\theta_2 & 0 & -\cos\theta_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^2 = A_3A_4 = \begin{bmatrix} \cos(\theta_3+\theta_4) & -\sin(\theta_3+\theta_4) & 0 & a_1\cos\theta_3 + a_2\cos(\theta_3+\theta_4) \\ \sin(\theta_3+\theta_4) & \cos(\theta_3+\theta_4) & 0 & a_1\sin\theta_3 + a_2\sin(\theta_3+\theta_4) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Therefore the forward kinematics of the Human Arm with Four DOF is described by

$$T_4^0 = T_2^0T_4^2 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & \mathbf{c}_x \\ r_{21} & r_{22} & r_{23} & \mathbf{c}_y \\ r_{31} & r_{32} & r_{33} & \mathbf{c}_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

$$\begin{aligned}
\mathbf{c}_x &= \cos\theta_1 \cos\theta_2[a_1 \cos\theta_3 + a_2 \cos(\theta_3 + \theta_4)] + \sin\theta_1[a_1 \sin\theta_3 + a_2 \sin(\theta_3 + \theta_4)] \\
\mathbf{c}_y &= \sin\theta_1 \cos\theta_2[a_1 \cos\theta_3 + a_2 \cos(\theta_3 + \theta_4)] - \cos\theta_1[a_1 \sin\theta_3 + a_2 \sin(\theta_3 + \theta_4)] \\
\mathbf{c}_z &= \sin\theta_2[a_1 \cos\theta_3 + a_2 \cos(\theta_3 + \theta_4)].
\end{aligned} \tag{5.1}$$

Substituting θ_2 by $(90° + \theta_2)$ into (5.1) yields:

$$\begin{aligned}
\mathbf{c}_x &= \sin\theta_1[a_1 \sin\theta_3 + a_2 \sin(\theta_3 + \theta_4)] - \cos\theta_1 \sin\theta_2[a_1 \cos\theta_3 + a_2 \cos(\theta_3 + \theta_4)] \\
\mathbf{c}_y &= -\cos\theta_1[a_1 \sin\theta_3 + a_2 \sin(\theta_3 + \theta_4)] - \sin\theta_1 \sin\theta_2[a_1 \cos\theta_3 + a_2 \cos(\theta_3 + \theta_4)] \\
\mathbf{c}_z &= \cos\theta_2[a_1 \cos\theta_3 + a_2 \cos(\theta_3 + \theta_4)].
\end{aligned} \tag{5.2}$$

Multiplying all the equations in system (5.2) by -1 yields:

$$\begin{aligned}
\mathbf{c}_x &= \cos\theta_1 \sin\theta_2[a_1 \cos\theta_3 + a_2 \cos(\theta_3 + \theta_4)] - \sin\theta_1[a_1 \sin\theta_3 + a_2 \sin(\theta_3 + \theta_4)] \\
\mathbf{c}_y &= \sin\theta_1 \sin\theta_2[a_1 \cos\theta_3 + a_2 \cos(\theta_3 + \theta_4)] + \cos\theta_1[a_1 \sin\theta_3 + a_2 \sin(\theta_3 + \theta_4)] \\
\mathbf{c}_z &= -\cos\theta_2[a_1 \cos\theta_3 + a_2 \cos(\theta_3 + \theta_4)].
\end{aligned} \tag{5.3}$$

Interchanging $\mathbf{c}_y$ and $\mathbf{c}_z$ into (5.3) yields:

$$\begin{aligned}
\mathbf{c}_x &= \cos\theta_1 \sin\theta_2[a_1 \cos\theta_3 + a_2 \cos(\theta_3 + \theta_4)] - \sin\theta_1[a_1 \sin\theta_3 + a_2 \sin(\theta_3 + \theta_4)] \\
\mathbf{c}_y &= -\cos\theta_2[a_1 \cos\theta_3 + a_2 \cos(\theta_3 + \theta_4)] \\
\mathbf{c}_z &= \sin\theta_1 \sin\theta_2[a_1 \cos\theta_3 + a_2 \cos(\theta_3 + \theta_4)] + \cos\theta_1[a_1 \sin\theta_3 + a_2 \sin(\theta_3 + \theta_4)].
\end{aligned} \tag{5.4}$$

5.7 Conclusions

In this chapter we have successfully applied our five-step methodology (chapter 3) to our second case study: *The Human Arm with Four Degrees of Freedom.* The kind of movements we obtained from this case study are *movements in 3D.* These movements are much more complex than those obtained from the previous case study. That is because, in this second case study we increased the number of degrees of freedom from two to four. Actually, the movements we obtained from this case study are very close to the movements performed by the left arm of a real human being.

After running several experiments with this case study, we realized that our learning algorithm performed pretty well and the resulting animation of this virtual creature was very real. A key fact of our learning algorithm was defining the reward function correctly.

In the following chapter we present a methodology for generating an *Alphabet of Movement* for *The Human Arm with Four Degrees of Freedom.*

Chapter 6

Towards an Alphabet of Movement

In this chapter we present a methodology for generating an *Alphabet of Movement* for the virtual creature presented in our second case study: *The Human Arm with Four Degrees of Freedom.* We start by giving a brief introduction, then we present our methodology and finally, in the last section, we show the experimental results obtained.

6.1 Introduction

In the preceding chapters 4 and 5 we presented two case studies. Now we use the experience obtained in such case studies to define a methodology for generating an *Alphabet of Movement* for the virtual creature presented in our second case study: *The Human Arm with Four Degrees of Freedom.*

In order to define such methodology, first we present some important definitions and previous knowledge.

6.1.1 Definition of an Alphabet of Movement

By *Alphabet of Movement* we mean:

- *a set of basic movements of a virtual creature which are stored in a Knowledge Base for later use.*

Generating this *Alphabet of Movement* is very important because it can be used as input to another learning or planning algorithm (working in a higher level) to obtain more complex animations of a virtual creature.

6.1.2 Previous Knowledge

Here we present some *previous knowledge* useful for defining our methodology:

1. The learning task of our second case study: *The Human Arm with Four Degrees of Freedom.* The task consists in learning to move the avatar's arm from an initial position $\mathbf{I_{x,y,z}}$ (end-effector's initial position) to a goal position $\mathbf{G_{x,y,z}}$ (end-effector's goal position).
2. Geometric figures such as plane curves with their respective equations (some of them expressed in parametric form).

6.1.3 Definition of a New Learning Task

Here we use the *previous knowledge* mentioned in section 6.1.2 to define a *new learning task.* Thus, our *new learning task* consists in:

- *The Human Arm with Four Degrees of Freedom learning to draw geometric figures in a plane* $z = c$.

This *new learning task* aims to emulate a person who is standing in front of a blackboard drawing a geometric figure on it. Perhaps a math teacher giving a plane geometry's lecture.

6.2 A Methodology for Generating an Alphabet of Movement

Based on the *new learning task* defined in section 6.1.3 and taking into account the *previous knowledge* mentioned in section 6.1.2, we propose the following methodology for generating an *Alphabet of Movement*:

1. Use the equation of a geometric figure to trace a path of n vertices in the perimeter of that figure.
2. For each pair of consecutive vertices v_i and v_{i+1}, call the learning task of our second case study to find a solution for those vertices. In other words, for each pair of consecutive vertices v_i and v_{i+1}, learn to move the avatar's arm from an initial position $\mathbf{I_{x,y,z}} = v_i$ (end-effector's initial position) to a goal position $\mathbf{G_{x,y,z}} = v_{i+1}$ (end-effector's goal position).

3. Merge all those particular $n-1$ solutions to get one general solution for the figure.

For a better understanding, we present a flow diagram of our methodology in Figure 6.1.

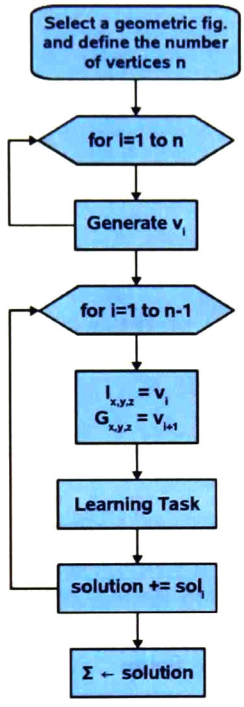


Figure 6.1: A Methodology for Generating an Alphabet of Movement.

If we repeat this methodology for all the different geometric figures we wish to draw, then we would get an *Alphabet of Movement* like the following:

$$\Sigma = \{\texttt{geometricFigure}_1, \texttt{geometricFigure}_2, \ldots, \texttt{geometricFigure}_n\}.$$

In the following section we present the result of applying our methodology to seven geometric figures. The *Alphabet of Movement* obtained from such geometric figures is also presented.

6.3 Experimental Results

Here we present the result of applying the methodology presented in section 6.2 to the following seven geometric figures:

1. Circle
2. Semicircle
3. A quarter of a circle
4. Horizontal line
5. Vertical line
6. Slash line
7. Backslash line

The results obtained for each one of these geometric figures is explained in detail in sections 6.3.1 to 6.3.7. Afterwards, we present in section 6.3.8 the *Alphabet of Movement* obtained from such geometric figures.

6.3.1 Circle

In order to generate this geometric figure, we applied the methodology presented in section 6.2 with the following two parameters:

1. Equation of a circle in parametric form:
   - $\begin{cases} x = r\cos\theta \\ y = r\sin\theta \end{cases}$ where: $r = 65$ and $0° \leq \theta \leq 360°$
2. Number of vertices:
   - $n = 13$.

After applying the methodology, we obtained the following solution:

$\texttt{circle} = 66 + 06 + 006 + 070 + 70 + 77 + 77 + 71 + 37 + 11 + 1661 + 6612.$

That is:

$\texttt{circle} = 66060060707077777713711166166 12.$

The animation generated from this solution is shown in Figure 6.2.

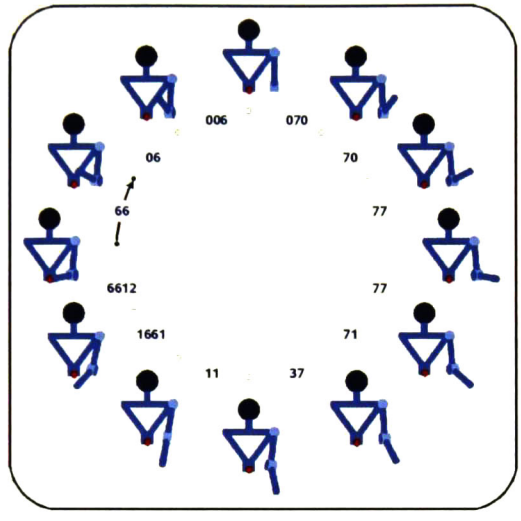


Figure 6.2: The Human Arm with Four DOF Drawing a Circle.

6.3.2 Semicircle

In order to generate this geometric figure, we applied the methodology presented in section 6.2 with the following two parameters:

1. Equation of a circle in parametric form:
   - $\begin{cases} x = r\cos\theta \\ y = r\sin\theta \end{cases}$ where: $r = 65$ and $0° \leq \theta \leq 180°$.
2. Number of vertices:
   - $n = 7$.

After applying the methodology, we obtained the following solution:

$\texttt{semicircle} = 66 + 06 + 006 + 070 + 70 + 77.$

That is:

$\texttt{semicircle} = 66060060707077.$

The animation generated from this solution is shown in Figure 6.3.

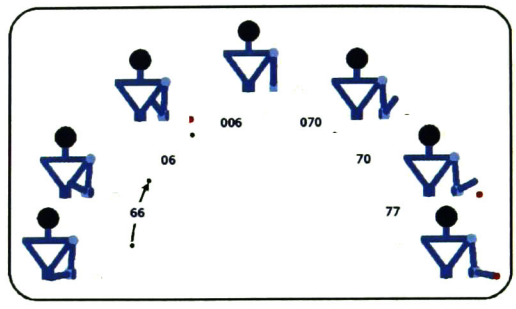


Figure 6.3: The Human Arm with Four DOF Drawing a Semicircle.

6.3.3 A Quarter of a Circle

In order to generate this geometric figure, we applied the methodology presented in section 6.2 with the following two parameters:

1. Equation of a circle in parametric form:
   - $\begin{cases} x = r\cos\theta \\ y = r\sin\theta \end{cases}$ where: $r = 65$ and $0° \leq \theta \leq 90°$.
2. Number of vertices:
   - $n = 4$.

After applying the methodology, we obtained the following solution:

`quartercircle` $= 66 + 06 + 006$.

That is:

`quartercircle` $= 6606006$.

The animation generated from this solution is shown in Figure 6.4.

6.3.4 Horizontal Line

In order to generate this geometric figure, we applied the methodology presented in section 6.2 with the following two parameters:

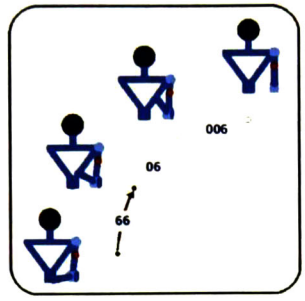


Figure 6.4: The Human Arm with Four DOF Drawing a Quarter of a Circle.

1. Equation of a horizontal line:
   - $y = c$, where: $c = -100$ and $length = 155$.
2. Number of vertices:
   - $n = 5$.

After applying the methodology, we obtained the following solution:

`hline` $= 0256 + 00 + 02 + 22747$.

That is:

`hline` $= 0256000222747$.

The animation generated from this solution is shown in Figure 6.5.

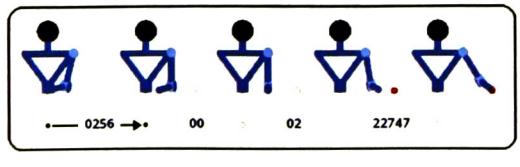


Figure 6.5: The Human Arm with Four DOF Drawing a Horizontal Line.

6.3.5 Vertical Line

In order to generate this geometric figure, we applied the methodology presented in section 6.2 with the following two parameters:

1. Equation of a vertical line:
   - $x = c$, where: $c = 55$ and $length = 155$.
2. Number of vertices:
   - $n = 5$.

After applying the methodology, we obtained the following solution:

$\texttt{vline} = 55 + 57 + 57 + 77$.

That is:

$\texttt{vline} = 55575777$.

The animation generated from this solution is shown in Figure 6.6.

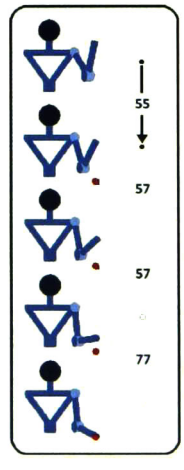


Figure 6.6: The Human Arm with Four DOF Drawing a Vertical Line.

6.3.6 Slash Line

In order to generate this geometric figure, we applied the methodology presented in section 6.2 with the following two parameters:

1. Equation of a slash line:
   - $y = x$, where: $length = 155$.
2. Number of vertices:
   - $n = 5$.

After applying the methodology, we obtained the following solution:

$\texttt{sline} = 5565 + 55 + 57 + 57$.

That is:

$\texttt{sline} = 5565555757$.

The animation generated from this solution is shown in Figure 6.7.

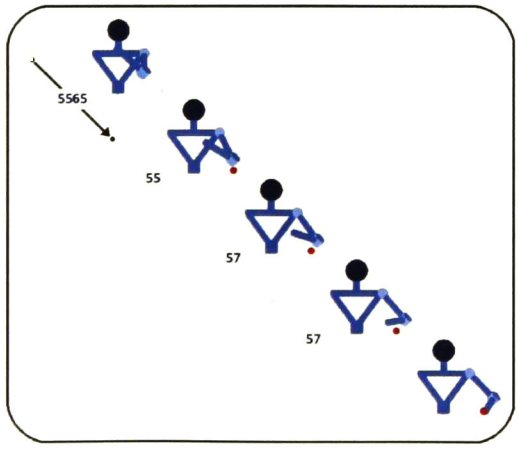


Figure 6.7: The Human Arm with Four DOF Drawing a Slash Line.

6.3.7 Backslash Line

In order to generate this geometric figure, we applied the methodology presented in section 6.2 with the following two parameters:

1. Equation of a backslash line:
   - $y = -x$, where: $length = 155$.
2. Number of vertices:
   - $n = 5$.

After applying the methodology, we obtained the following solution:

$\texttt{bsline} = 5156 + 1517 + 17 + 717.$

That is:

$\texttt{bsline} = 5156151717717.$

The animation generated from this solution is shown in Figure 6.8.

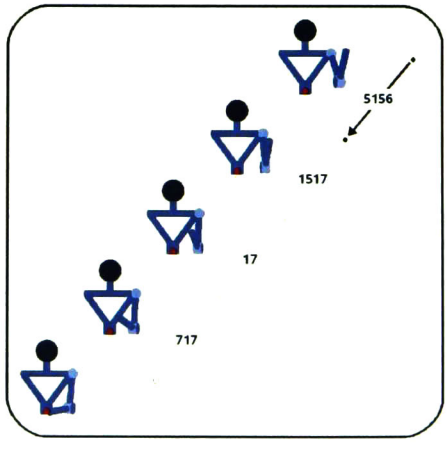


Figure 6.8: The Human Arm with Four DOF Drawing a Backslash Line.

6.3.8 Alphabet of Movement

Here we present the *Alphabet of Movement* generated after applying the methodology presented in section 6.2 to the seven geometric figures presented in sections 6.3.1 to 6.3.7.

Therefore, the *Alphabet of Movement* generated is

- Σ = { `circle`, `semicircle`, `quartercircle`, `hline`, `vline`, `sline`, `bsline` }, where:
  - `circle` = 66060060707077777371116616612.
  - `semicircle` = 66060060707077.
  - `quartercircle` = 6606006.
  - `hline` = 0256000222747.
  - `vline` = 55575777.
  - `sline` = 5565555757.
  - `bsline` = 5156151717717.

That is, the *Alphabet of Movement* generated is

- Σ = { 66060060707077777371116616612, 66060060707077, 6606006, 0256000222747, 55575777, 5565555757, 5156151717717 }.

6.4 Conclusions

In this chapter we have presented a methodology for generating an *Alphabet of Movement* for *The Human Arm with Four Degrees of Freedom.* The *Alphabet of Movement* generated is very important because it can be used as input to another learning or planning algorithm (working in a higher level) to obtain more complex animations of our virtual creature.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis we have presented a novel methodology for the *Animation of Virtual Creatures Using Learning.* Our methodology proposes the use of a well-known *Reinforcement Learning* algorithm such as *Q-learning* (Watkins 1989).

We presented two case studies in which we have successfully applied our methodology obtaining encouraging results:

- *The Human Arm with Two Degrees of Freedom*, and
- *The Human Arm with Four Degrees of Freedom.*

For both case studies we ran several experiments in which we realized that our learning algorithm performed pretty well and the resulting animation was very real.

Although our methodology has proved to work well for these case studies, its success depends enormously on how well we defined the *reward function.*

We also presented a methodology for generating an *Alphabet of Movement* for *The Human Arm with Four Degrees of Freedom.* In this methodology, we used the *experience* obtained in our case studies and some *previous knowledge* like *plane geometry equations* to generate the following *Alphabet of Movement*:

$\Sigma = \{$ `circle, semicircle, quartercircle, hline, vline, sline, bsline` $\}$

This methodology besides generating an *Alphabet of Movement*, it defined a *new learning task*:

- *The Human Arm with Four Degrees of Freedom learning to draw geometric figures in a plane $z = c$.*

This *new learning task* opened the path for our future work which is presented in the following section.

7.2 Future Work

7.2.1 Short Term Goals

Our short term goals are:

- To use the current *Alphabet of Movement* as input to another learning or planning algorithm (working in a higher level) to obtain more complex animations of our current virtual creature.
- To increase the number of degrees of freedom of our current virtual creature, perhaps by adding the left hand.
- To add more extremities to our current virtual creature in order to obtain a learning task much more complex; a learning task that involves the use of more than one extremity like giving an applause or handing a baseball bat.

7.2.2 Long Term Goals

Our long term goals are:

- To use the current *Alphabet of Movement* to define a new learning task for our second case study like: *learning to write a word using the alphabet in a plane $z = c$.*
- To propose a new case study with a new virtual creature and with more degrees of freedom in which we can apply our methodology.

Bibliography

[1] Developer Resources for Java Technology. `http://java.sun.com/`.

[2] Java 3D Parent Project. `https://java3d.dev.java.net/`.

[3] Toni Conde, William Tambellini, and Daniel Thalmann. Behavioral Animation of Autonomous Virtual Agents Helped by Reinforcement Learning. In *Proceedings of the 4th International Working Conference on Intelligent Virtual Agents, IVA 2003*, pages 175–180, 2003.

[4] Toni Conde and Daniel Thalmann. An Artificial Life Environment for Autonomous Virtual Agents with multi-sensorial and multi-perceptive features. *Computer Animation and Virtual Worlds*, 15(3-4):311–318, 2004.

[5] Toni Conde and Daniel Thalmann. Autonomous Virtual Agents Learning a Cognitive Model and Evolving. In *Proceedings of the 5th International Working Conference on Intelligent Virtual Agents, IVA 2005*, pages 88–98, 2005.

[6] Toni Conde and Daniel Thalmann. Learnable Behavioural Model for Autonomous Virtual Agents: Low-Level Learning. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2006*, pages 89–96, 2006.

[7] C. Gatzoulis, W. Tang, and T. R. Wan. Fuzzy Reinforcement Learning for An Evolving Virtual Servant Robot. In *Proceedings of the 2004 IEEE International Workshop on Robot and Human Interactive Communication, ROMAN 2004*, pages 685–690, 2004.

[8] John H. Holland, Lashon B. Booker, Marco Colombetti, Marco Dorigo, David E. Goldberg, Stephanie Forrest, Rick L. Riolo, Robert E. Smith, Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson. What is a Learning Classifier System? In Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson, editors, *Learning Classifier Systems. From Foundations to Applications*, volume 1813 of *LNAI*, pages 3–32, Berlin, 2000. Springer-Verlag.

[9] Michal Jakob, Jan Tožička, and Michal Pěchouček. Collaborative Learning with Logic-Based Models. In *Proceedings of Adaptive Learning Agents and Multi-Agent Systems, ALAMAS 2007*, pages 89–103, 2007.

[10] Tsai-Yen Li and Yang-Chuan Shie. An Incremental Learning Approach to Motion Planning with Roadmap Management. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002*, pages 3411–3416, 2002.

[11] Ryszard S. Michalski. LEARNABLE EVOLUTION MODEL: Evolutionary Processes Guided by Machine Learning. *Machine Learning*, 38(1-2):9–40, 2000.

[12] Tom M. Mitchell. *Machine Learning.* McGraw Hill, 1997.

[13] Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Reinforcement Learning for Humanoid Robotics. In *Proceedings of the 3rd IEEE-RAS International Conference on Humanoid Robots, ICHR 2003*, 2003.

[14] Andrew Scholer, Richard Angros, Jr., Jeff Rickel, and W. Lewis Johnson. Teaching Animated Agents in Virtual Worlds. In *Smart Graphics: Papers from 2000 AAAI Spring Symposium*, pages 46–52, 2000.

[15] Mark W. Spong and M. Vidyasagar. *Robot Dynamics and Control.* John Wiley & Sons, Inc., 1989.

[16] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[17] Daniel Torres and Pierre Boulanger. The ANIMUS project: a framework for the creation of interactive creatures in immersed environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST 2003*, pages 91–99, 2003.

[18] W. Zhang and T. G. Dietterich. A Reinforcement Learning Approach to Job-shop Scheduling. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1114–1120, 1995.

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Animation of Virtual Creatures Using Learning

del (la) C.

Moisés UC CETINA

el día 30 de Mayo de 2008.

Dr. Luis Ernesto López Mellado
Investigador CINVESTAV 3B
CINVESTAV Unidad Guadalajara

Dr. Félix Francisco Ramos Corchado
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

Dr. Jérôme Leboeuf Pasquier
Profesor Investigador
Universidad de Guadalajara - Centro Universitario de Ciencias Exactas e Ingeniería