Control Behavior of 3D Humanoid Animation Object Using Reinforcement Learning

Yuesheng He

Abstract

The ability to learn is a potentially compelling and important quality for interactive 3D human avatars or virtual humans. To that end, we describe a practical approach to real-time learning for 3D virtual humans. Our implementation is grounded in the techniques of reinforcement learning and informed by insights from avatar's behavior training. It simulates the learning task for characters by enabling them to take advantage of predictable regularities in virtual graphical world and allowing them to make maximal use of any supervisory signals. We built an autonomous animated virtual human that can be trained with a technique used to train proper actions in the real world. Capabilities demonstrated include being trained to recognize and use shapes of 3D models' patterns as cues for actions, as well as to synthesize new actions from novel paths through its motion space. A key contribution of this paper is to demonstrate that by considering the three aspects on the problem of state, action, and reaction space discovery at the same time, by forming a value function on a manifold, the solution for each becomes easier. Finally, we articulate heuristics and design principles that make learning practical for synthetic animation human characters.

1 Introduction

We believe that interactive synthetic characters must learn from experience if they are to be compelling over extended periods of time. Furthermore, they must adapt in ways that are immediately understandable, important and ultimately meaningful to the virtual environment interacting with it. Computer system provides an excellent example of systems that do just this: 3D graphical human-like avatar[3][6].

Remarkably, virtual human do the actions with the effort of simulating our behavior, but little understanding of context and legend beyond their use as cues in the graphical environment. In addition, they are only able to learn causality if the events, actions and consequences are proximate in space and time, and as long as the consequences are motivationally significant[6].

Nonetheless, the learning avatars do allow them to behave human-like common sense and ultimately exploit the highly adaptive ability to different kinds of 3D environments.

Our belief is that by embedding the kind of learning of which avatars are capable into synthetic characters, we can provide them with an equally robust mechanism for adapting in meaningful ways to act like people in the 3D virtual environment which they are interacting.

In this paper, we describe a practical approach to real-time learning for 3D avatar characters that allows them to learn the kinds of things that people seem to learn so easily[7].

We ground our work in the machine learning theory of reinforcement learning, in which a creature learns to maximize reward in the absence of a teacher. Additionally, our approach is informed by insights from robot training, where a teacher is available. Robots and their trainers act as a coupled system to guide the robots exploration of its state, action, and state-action spaces . Therefore, we can simplify the learning task for autonomous animation characters by (a) enabling them to take advantage of predictable regularities in their world, (b) allowing them to make maximal use of any supervisory signals, either explicit or implicit, that the world offers, and (c) making them easy to train by interacting with virtual environment. Using this method, we implemented the autonomous animated avatar that can be trained with a technique used to train and behave like human. The synthetic avatar thus mimics some of a real human's ability to learn including:

- The best action to perform in a given context. What form of a given action is most reliable in producing reward;
- The relative reliability of its actions in producing a reward and altering its choice of action accordingly;
- To recognize new and valuable contexts such as shapes' patterns[1];

- To synthesize new and reasonable actions the training.

In order to accomplish these learning tasks, the system must address the three important problems of state, action and state-action space discovery. A key contribution of this paper is to show how these processes may be addressed in an integrated approach that guides and simulating the human-like processes. We emphasize that our behavioral architecture is one in which learning can occur, rather than an architecture that solely performs learning. As we will see, learning has important implications for many aspects of a general behavior architecture, from the design of the perceptual mechanism to the design of the 3D animation graphical system. Conversely, careful attention to the design of these components can dramatically facilitate the learning process. Hence, an important goal of this paper is to highlight some of these key design considerations and to provide useful insights apart from the approach that we have taken. We begin by surveying the framework of the approach. We then turn to a discussion of reinforcement learning. We introduce the core concepts and terminology, discuss the application of reinforcement learning to training avatar characters, and take on insights into the trial and error process. We then describe our approach, reviewing our key representations and processes for state, action and state-action space discovery. We present our experiment with different terrains, our virtual human, and discuss limitations of our approach. We conclude with a summary of what we see as the key lessons from our work.

2 Overview of The Framework

Motion action context is extensively used in computer animation, because it is able to describe all the subtleties of real human motion. By piecing together many some different type of short motion clips, we can further create novel but realistic motions. Consequently,the way of arranging the clips of motions to achieve specific goals is an important research topic[12].

A number of algorithms have been developed to represent plausible transitions between motion clips with graph structures. With these techniques, novel motions can be generated simply by building walks on the graph. For off-line applications, where the full motion specification is known in advance, a global sub-optimal or close-to-optimal solution that minimizes an objective function, such as a certain energy, can be found. In interactive applications, new input is continuously arriving and the decision for selecting the next clip needs to be made in a very short amount of time.

Therefore, only local search can be performed to generate motions in response to the dynamic 3D environments. The challenge for local search methods is to synthesize motions that require planning. Motion planning is important to achieve realistic results in many scenarios. For example, one may need to prepare well in advance to grasp an object at a particular location. Hence, first of all is to plan a total strategy of the action, then instead of trying to search a point-to-point path on the graph, we are going to face a dynamic changing 3D environment. Thus reinforcement learning techniques is used to train a motion controller off-line, which can make on-line decision quickly in any given situation. Some methods have been proposed in computer animation to utilize reinforcement learning to obtain policies for choosing actions that will increase long term expected rewards.

Thus, the whole motion can be separate into different clips. i.e. if the avatar want to go through a terrain full of obstacles, its motion can be constructed as Figure1shows. The state space and use dynamic programming to construct

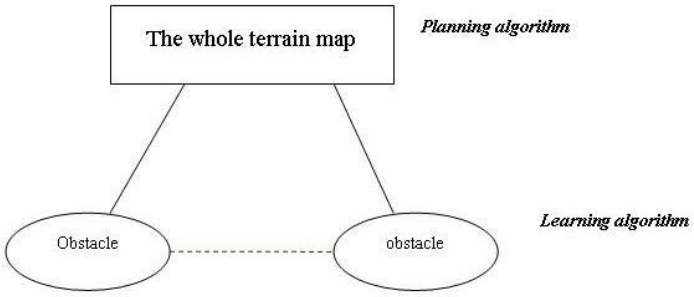

Figure 1. The framework of the the whole procedure of a virtual human's action in the 3D graphical environment.

a sample-based value function for actions.

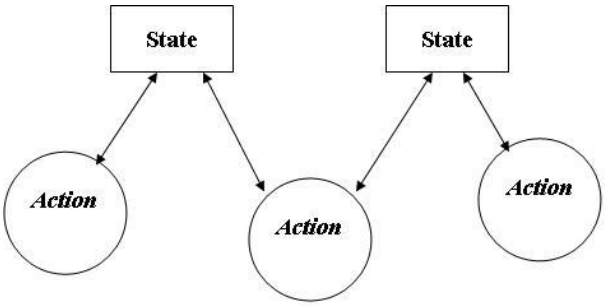

Figure 2. Every motion of a virtual human in the 3D graphical environment can be treated as a Semi-Markov Decision Process.

We will consider the action as a general class of random processes with trajectories in a metric space which are

	$DynamicProgramAlgorithm$
1	Q.Insert(Gx) and mark Gx as visited
2	while Q not empty do
3	x ← Q.GetFirst() //Get the priori queue
4	return SUCCESS
5	forall u-1 ∈ U-1(x)
6	x ← f-1(x', u-1)
7	if x not visited
8	Mark x as visited
9	Q.Insert(x)
10	else
11	Resolve duplicate x
12	return FAILURE

Table 1. The dynamic programming algorithm to update the value of action's policy.

continuous from the right and have limits from the left at any point of the half-line. These processes were continuous semi-Markov processes in according to the property of their first exit streams. The structure of the these processes of actions are showed on Figure 2 Thus, it is necessary to develop methods or to modernize traditional methods of investigation, which do not use the simple Markov property[12].

We can support value functions with to evaluate the actions' policy. Moreover, the above methods suffer from the limitation that the space of available motions is discrete. The parametric space is an abstract space defined by kinematic or physical attributes of motions. By parameterizing all motion samples in the space, and by blending among multiple motions, motion interpolation can create novel motions that have specific kinematic or physical attributes. We build a continuous parameterized motion space for similar motions that provide efficient control for interpolation.

To improve the action's policy π episode by episode, we adopt a value function Q and dynamic programm algorithm to achieve the goal Table 1[21].

To improve motion interpolation with the use of geometric statistics, treating interpolation as statistical prediction of missing data in the parametric space. We analyze interpolated human motions for physical correctness and show that the interpolated results are close to the physically correct motions. Cooper et al. proposed active learning to adaptively sample the parametric space so that the space can be well sampled with a reduced number of clips . Recently, researchers have also combined motion graphs with parametric synthesis to form richer, more complete motion spaces. In order to provide proper control, we present a way to learn parametric motion controllers, which can compute near-optimal parameters for motion synthesis for the 3D graphical environment in real-time.

3 Learning Algorithm on Manifold

In this section we describe a reinforcement learning framework to obtain motion controllers for interactive character animation. Using a database of atomic motion clips, our goal is to generate natural character motion as a sequence of clips. At each time step, the motion controller decides which motion clip best follows the user input and respects constraints imposed by the environment. This decision must be made quickly, since time lags are not allowed in interactive environments. The controller should also be able to achieve user objectives that require planning ahead of time. In addition, both user input and the environment should be represented using continuous parameters to allow for proper control.

We begin to introduce the Markov decision process (MDP) model for the control of the virtual human's action, and describe methods for approximately solving MDPs. This section is principally about choosing a basis for approximation of real-valued functions called value functions, which are central to solving Markov decision processes [13]. For in the most cases, previous work has modeled value functions as vectors in a Euclidean space to evaluate the action's policy π. One of the novel ideas is the approach to treat value functions as elements of a vector space on a graph or manifold [12]. This approach enables constructing basis functions that capture the large-scale (irregular) topology of the graph, and approximating (value) functions on these graphs by projecting them onto the space spanned by these bases.

A discrete Markov decision process (MDP) $M = (S, A, P^a_{s,s'}, R^a_{s,s'})$ is defined by a finite set of discrete states S, a finite set of actions A, a transition model $P^a_{s,s'}$ specifying the distribution over future states s' when an action a is performed in state s, and a corresponding reward model $R^a_{s,s'}$ specifying a scalar cost or reward [13].

Any optimal policy π defines the same unique optimal value function V which satisfies the nonlinear constraints:

$$V(s) = \max_a (R^a_{s,s'} + \gamma \sum_{s' \in S} P^a_{s,s'} V(s'))$$

where $Rsa = \sum_{s' \in S} P^a_{s,s'} R^a_{s,s'}$ is the expected immediate reward.

The Bellman operator T_π on the space of value function which is used to evaluate the policy π can be written as

$$T^\pi(V) = R_{s\pi(s)} + \gamma \sum_{s'} P^{\pi(s)}_{ss'} V(s')$$

Thus, the value function Vp associated with following a (deterministic) policy p can be defined as

$$V^\pi(s) = T(V^\pi(s)) = R_{s\pi(s)} + \gamma \sum_{s' \in S} P^{\pi(s)}_{ss'} V^\pi(s').$$

As it will be explained in the following part,in order to find the best policy π, we pose an optimization problem where the discrete values of temporal states are the variables. The goal of the optimization problem is to maximize the value function which requires determined by the states and actions, while enforcing the user's constraints.

Therefore, the question can be treated as a learning problem on a Riemannian manifold. There is a rich and well-developed theory of the Laplace operator on manifolds, which we can only briefly summarize here. The Laplace-Beltrami operator has been extensively studied in the general setting of Riemannian manifolds [16]. Riemannian manifolds have been actively studied recently in machine learning in several contexts, namely in the context of designing new types of kernels for supervised machine learning[17] and faster policy gradient methods using the natural Riemannian gradient on a space of parametric policies [18][19][20].

Although a full discussion of these perspectives is beyond this paper, they are worth noting in order to gain deeper insight into the many remarkable properties of the Laplacian.

The Laplacian is defined as

$$\Delta = divgrad = \frac{1}{\sqrt{\det g}} \sum_{i,j} \partial_i (\sqrt{\det g^{ij}} \partial_j)$$

where div and grad are the Riemannian divergence and gradient operators.

constraints in our optimization process. Detail vs. volume preservation. It is well known that the details of a shape at a point in space are preserved during a deformation if the local transformation that point undergoes is close to rigid.

To test our reinforcement learning algorithm's efficiency, we used a simulating environment in Matlab software.

This environment is based on Sutton and Barto's (1998)[21] Mountain Car Task as described in "Reinforcement Learning: An Introduction". This simulation program is running on the Matlab 7.02b as Figure 3 shows.

Consider a task where an agent must drive an underpowered car up a steep mountain road. Since gravity is stronger than the cars engine, even at full throttle the car cannot simply accelerate up the steep slope. The cars movement is described by two continuous output variables, position and velocity, and one discrete input representing the acceleration of the car.

In this application the state space S is naturally embedded in R_d as a sub-manifold, or some other type of geometrically structured subset. The shape of the manifold is showed in Figure 4.

The training used Q-learning algorithm in which the $\varepsilon - greedy$ policy should be used. We will randomly choose a 100 episode table to test the algorithm. The episodes will be truncated at 1000 time steps.

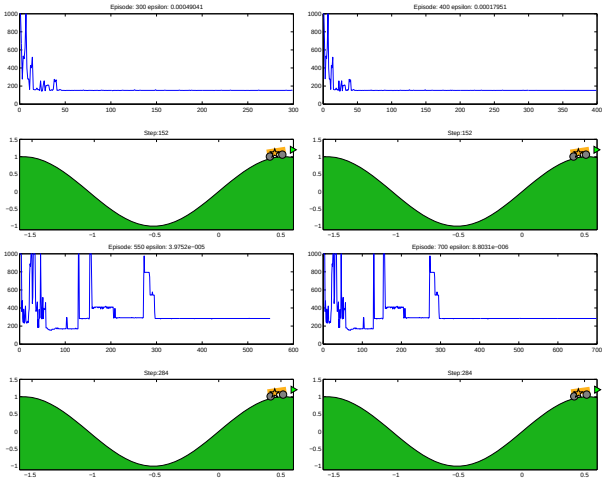

Figure 3. The clips of episodes of reinforcement learning training for the Mountain Car on the simulating software of Matlab. The up axes are representing the episode and steps and the down one is the animation clip of the simulation.

We used a Laplacian matrix as a window-based mask, i.e. it relies on a convolution mask to perform temporal filtering. We subtracted off all low frequency components from the original proto-value Qtable. Then we used high frequency edge descriptions to enhance the Qtable and compute a improved Qtable.

At the beginning, the Q table is all zero. After 100 episode training, we record a table. This table will be used in the two branches. One just keep on training without any change, the other will be transformed by laplace operator to improve the performances. The later's manifold shape shows in Figure 5.

Experimental results and comparisons with state-of-the-art methods are presented in Figure6 and Figure7. We will conclude all the result with a discussion and some future research directions in Section 5. In this section, we can make a conclusion that with the transformation for improving the Qtable, the improved one has a more stable and less steps training than original one. Moreover, the improved one's rewards are also better than original one.

4 Experimental Result

Our 3D graphical platform is the open source software : Delta3D.

Delta3D is an Open Source engine which can be used for games, simulations, or other graphical applications. Its modular design integrates other well-known Open Source

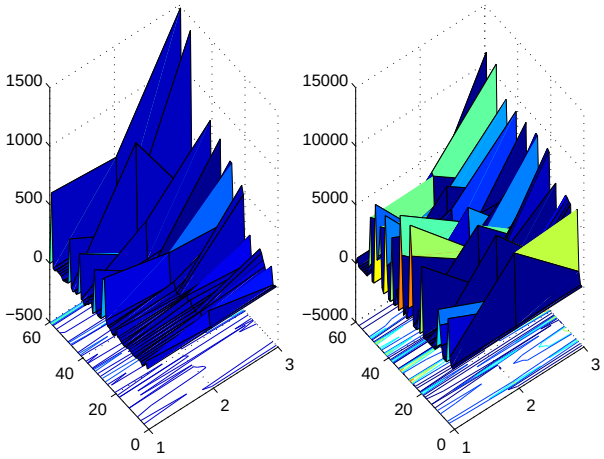

Figure 4. Qtable - The left one is the value function or Q function's manifold after 100 episode training, and the right one is it after 300 episode training.

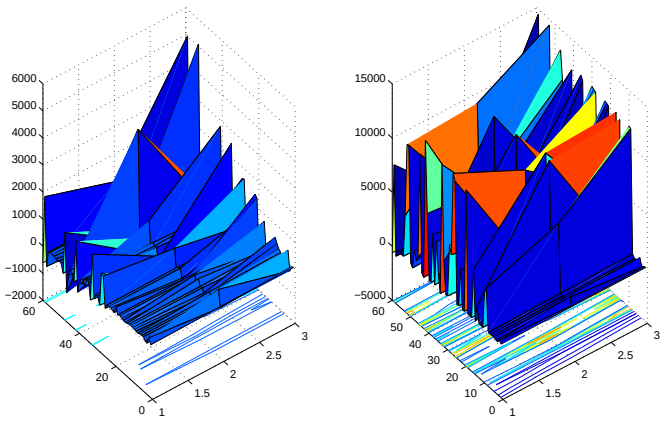

Figure 5. The Qtable Laplacian - The left one is the value function or Q function's manifold after 100 episodes training and laplacian improvement, and the right one is it after 300 episode training.

projects such as Open Scene Graph(OSG), Open Dynamics Engine(ODE), Character Animation Library (CAL3D), and OpenAL. Rather than bury the underlying modules, Delta3D just integrates them together in an easy-to-use API – always allowing access to the important underlying components. This provides a high-level API while still allowing the end user the optional, low-level functionality. Delta3D renders using OpenGL and imports a whole list of diverse file formats (.flt, .3ds, .obj, etc.).

Delta3D is released under the GNU Lesser General Public License (LGPL). The underlying modules may have their own licensing, but are at the minimum, considered Open Source and are freely distributable.

We used two virtual humans : "Cally" and "Paladin". Where they have a different ability and will behave differently to interact with the environments.

The task of the both avatars is to access the object (i.e. a beast), then interacts with it (i.e. shot a arrow to it).

The Cally and Paladin has different acting abilities and different absorb states which has been showed in Figure8, Figure9,Figure10 and Figure11.

The task of Cally and Paladin in a town is to access different kind of point to get reward. Cally can walk or jog to the object which show in the Figure12, and has two absorbing states which show in the Figure13. Paladin can walk, jog or sneak to the object which show in the Figure14, and has one absorbing state which shows in the Figure15.

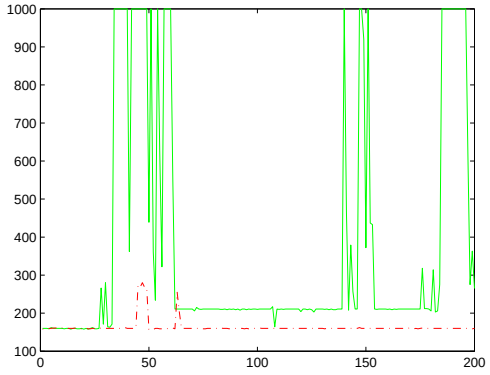

Figure 6. The steps (maximum is 1000) of training the Mountain Car. The green line is represented the Q function without any enhancement. And the red one is represented the Q table has been filtered by laplacian operator.

5 Conclusion and Discussion

This article presents a method for creating virtual human in the 3D graphical environment with learning ability on 3D animation. The example takes different 3D avatars which have different abilities to show different performance as the environment is dynamically changing. The result of blending actions animation presents that the virtual human has a more human-like behavior ,moreover intelligent ability to adapt the environment by learning ability.

We constructed a framework of controlling the motion of 3D human-like avatar by treating it as a Semi-Markov Decision Process.

By researching the reinforcement learning, we are able to choose the appropriate method to control the action of 3D human-like avatar. Besides, to treat the value function as a manifold which depends on the states and actions is proved to be a effective way to improve the Q function in the training process.

There are many opportunities to improve the techniques presented here. First, according to the shapes of manifold, there will be many powerful method to improve the performance of the learning algorithm. Second, if the virtual human can independently recognizes or classifies the objects and behaves in different way according to them, the system will be much better.

In this case, instead of making the animations frame by frame, the intelligent 3D avatar would be a powerful tools to achieve the work. Moreover, a more effective machine learning algorithm is one the key part of this area.

References

[1] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin,Shape distributions, *ACM Transactions on Graphics*, Vol. 21, No. 4, October 2002, Page 807—832.

[2] Z.M. , D. Reidsma, A. Nijholt, Human Computing, Virtual Humans and Artificial Imperfection, *ICMI'06*, Page 179—184.

[3] Norman I.Badler, Cary B.Phillips, Bonnie Lynn Webber,*Simulating Humans: Computer Graphics Animation and Control*, 1993.

[4] T. Conde, D. Thalmann, An Integrated Perception for Autonomous Virtual Agents: Active and Predictive Perception, Computer Animation and Virtual Worlds, Volume 17, Issue 3-4, John Wiley, 2006

[5] Moccozet L, Thalmann N. M., Dirichlet Free-Form Deformation and their Application to Hand Simulation[J], *Proceedings Computer Animation97*, IEEE Computer Society, 1997, Page 93—102.

[6] Catherine Zanbaka1, Amy Ulinski, Paula Goolkasian, Larry F. Hodges, Social Responses to Virtual Humans: Implications for Future Interface Design, *CHI 2007 Proceedings*, Page 1561-1570.

[7] Edward M. Sims, Reusable, lifelike virtual humans for mentoring and role-playing, *Computers & Education*, 49 2007, Page 75-92.

[8] Lucio Ieronutti , Luca Chittaro, Employing virtual humans for education and training in X3D/VRML worlds, *Computers & Education*, Page 93-109.

[9] Weixiong Zhang, Randall W. Hill, Jr., A Template-Based and Pattern-Driven Approach to Situation Awareness and Assessment in Virtual Humans, *Agents*, 2000, Page 116—123.

[10] Z.M. , D. Reidsma, A. Nijholt, Human Computing, Virtual Humans and Artificial Imperfection, *ICMI'06*, Page 179—184.

[11] Sridhar Mahadevan: Fast Spectral Learning using Lanczos Eigenspace Projections. AAAI 2008, pp. 1472-1475.

[12] Sridhar Mahadevan and Mauro Maggioni, "Proto-Value Functions: A Laplacian Framework for Learning Representation and Control in Markov Decision Processes" , Journal of Machine Learning Research, pp. 2169—2231, vol. 8, 2007, MIT Press.

[13] M. L. Puterman. Markov decision processes. Wiley Interscience, New York, USA, 1994.

[14] A. Barto, and S. Mahadevan. Recent Advances in Hierarchical Reinforcement Learning. DiscreteEvent Systems: Theory and Applications, 13:41—77, 2003.

[15] F. Chung. Spectral Graph Theory. Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1997.

[16] S Rosenberg. The Laplacian on a Riemannian Manifold. Cambridge University Press, 1997.

[17] J. Lafferty and G. Lebanon. Diffusion kernels on statistical manifolds. Journal of Machine Learning Research, 6:129—163, 2005.

[18] S. Kakade. A Natural Policy Gradient. In Proceedings of Neural Information Processing Systems. MIT Press, 2002.

[19] J. Bagnell and J. Schneider. Covariant policy search. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pages 1019—1024, 2003.

[20] J. Peters, S. Vijaykumar, and S. Schaal. Reinforcement learning for humanoid robots. In Proceedings of the Third IEEE-RAS International Conference on Humanoid Robots, 2003.

[21] Sutton, R. S., and Barto, A. G. Reinforcement Learning C An Introduction. MIT Press, 1998.

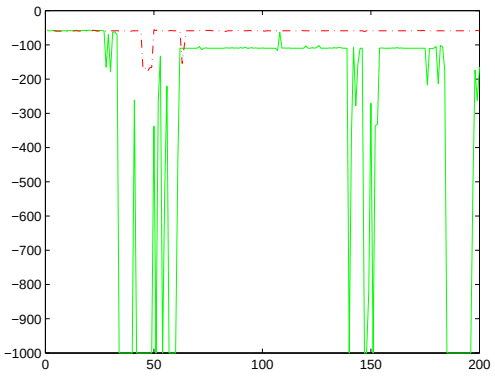

Figure 7. The rewards (every step without reaching the goal is -1, and with reaching the goal is 100) of training the Mountain Car. The green line is represented the value of Q function without any enhancement. And the red one is represented the Q table has been filtered by laplacian operator.

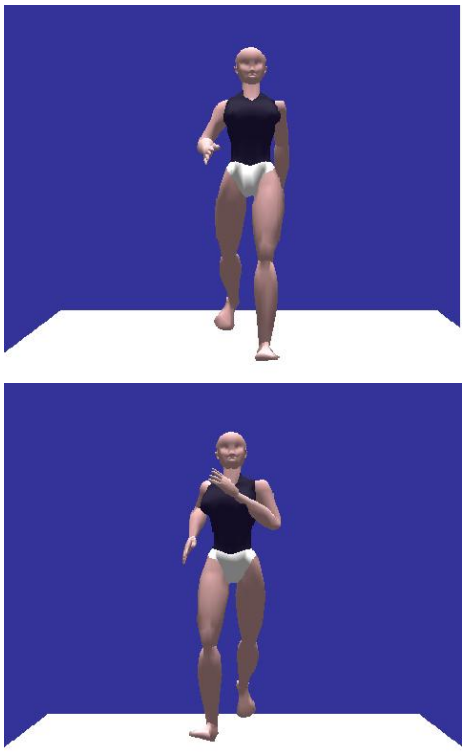

Figure 8. Cally has abilities of walk and jog to access the object. She has two different kinds of actions.

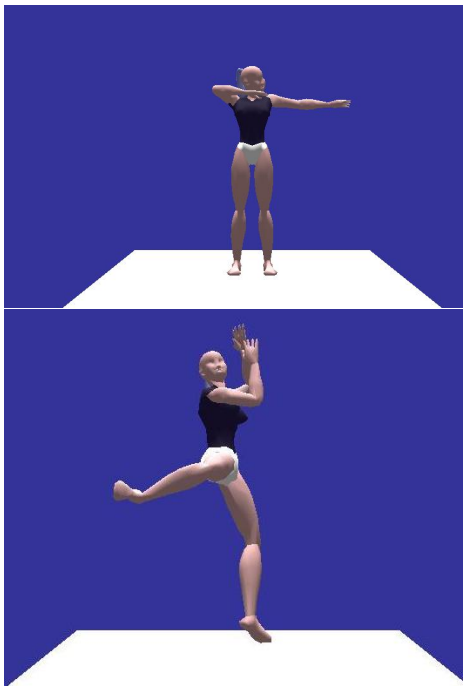

Figure 9. Cally has abilities of shotting an arrow and kicking the object. She has two different absorbing states.

Figure 10. Paladin has abilities of walk, jog and sneak to access the object. He has Three different actions.

Figure 11. Paladin has a ability of shotting the object. He has one absorbing state.

Figure 12. Cally 's typical action to access the object point in the town.

Figure 13. Cally chooses the action when reach the object point, if it is far, then she shot, otherwise, she will kick.

Figure 14. Paladin's typical action to access the object point in the town. Where the object is far, he walks, then he jogs, after that, he sneaks

Figure 15. Paladin reaches the proper position, then begin to shot.