Dynamic Autonomous Agents: Game Applications

Siome Goldenstein, Edward Large and Dimitris Metaxas

Center for Human Modeling and Simulation
Computer and Information Science Department
University of Pennsylvania, Philadelphia, PA, USA 19104-6389
siome,large,dnm@graphics.cis.upenn.edu

Abstract

We present a method for the generation of real-time dynamic autonomous agents in game environments. The method is based on the use of dynamical systems theory which allows us to express intelligent behaviors using systems of differential equations. These differential equations operate at two distinct levels. At one level, the differential equations governing movement comprise a carefully designed set of attractor-repeller fields. At a second level, behavior selection is modeled using a competition dynamics that determines the relevant contributions to the movement dynamics at each time instant. Thus, our autonomous agents are capable of exhibiting useful behaviors in complex environments. Using this approach we are able to demonstrate in real time behaviors such as single and multiple target tracking with both stationary and moving obstacles.

Keywords: *Digital Agents, Game Animation, Motion Planning, Dynamical Systems.*

1. Introduction

The importance of game and simulation applications grows everyday, as does the need for animated agents that operate autonomously in these environments. These agents must be able to exhibit certain behaviors without user intervention. Various methods for generating higher levels of behavior and movement decisions were investigated first in the pioneering work by Reynolds [19], and then in work by others [8, 27, 18, 3, 14, 20, 9, 25, 13]. *AI* approaches [8, 11] are capable of generating autonomous behavior, but such techniques typically require complex inferencing mechanisms. This may require considerable computational resources, raising the question of scaling up such systems as the number of independent agents grows, or when each agent has a completely different goal and behavioral directives. In addition to these factors, agents must be able to interact with real-time moving objects that might either contribute to or compromise the final goal. Other approaches to this problem employ computational geometry techniques in rather restricted environments [26], while learning, perception, and dynamic techniques have also been employed. [21, 25, 13, 7]. Dynamic system approaches to this problem have mostly been explored at the level of control theory [6, 28, **?**, 25] which restricts the type of behaviors that can be simulated.

In this paper, we investigate and develop an alternative methodology that has its roots in behavior-based robotics (e.g., [4, 5]) and is based on a novel way of combining differential equations exhibiting particular behaviors. According to this methodology, one defines a representation whose dimensions correspond to agent behavior. Using this type of approach, Schöner and colleagues have developed a dynamical system for governing robot movement. In this system a set of behavioral variables, namely heading direction and velocity, defines a state space in which a dynamics of robot behavior is described [22, 23]. Movement is governed by a nonlinear dynamical system that generates a time course of the behavioral variables. The system dynamics are specified as a nonlinear vector field, while the task that the agent will execute depends upon the task constraints. Task constraints are modeled as component forces, defining attractors and repellers of the dynamical system. The individual constraint contributions are additively combined into a single vector field, which determines the observed behavior.

This early approach to the autonomous generation of movement was restricted to the generation of individual behaviors such as navigation toward a fixed goal. Recently, however, methods have been developed to allow an agent to arbitrate among a large number of potential behaviors, and to generate complex sequences of activity in a manner that is robust, yet flexible in the face of a changing environment [24, 10]. To achieve this result a second dynamical system is defined that operates in the space of task constraints. This dynamic approach forces task constraints to compete for representation at the behavioral level. Thus, at any given

time the behavioral vector field (and the observed behavior) comprises a subset of possible task constraints. The parameters of the dynamical system are chosen in such a way that the agent's behavior is appropriate to the current situation.

Here we adapt the above methodology to develop autonomous dynamic behaviors for games. In particular, we devise a set of time adaptive differential equations to rule the heading angle and forward speed of a given digital autonomous agent. Based on a principled combination of these equations we create a set of relatively complex low-level behaviors which are reactive in nature. Using this system, decisions are made on-line and do not require any previous memory, training or global planning. The set of targets and obstacles can change during the course of the simulation, since the agent is able to make "smart" local decisions based on its current global knowledge of the dynamic environment it is situated. An example of such a behavior is that the agent will temporarily disregard a target if there is an unsurpassable moving or stationary obstacle immediately between them. It will then focus (as a human would) on first avoiding the obstacle and then it will refocus on the target.

Our system allows single or multiple target tracking in the presence of multiple static or moving obstacles. The design of the differential equations allows the tracking of targets whenever their position is within the visible cone of an agent requiring only the estimation of its current position. However, obstacles are processed in a local fashion based on their relative location to the agent and the target. Given our applications, in our current implementation our agents are memoryless and reactive in nature and depending on the situation (emergence of new obstacles and/or targets) their behavior can change abruptly.

In the following sections, we present previous related work in the area, the design of our system and the series of real-time experiments geared towards game applications.

2. Movement Dynamics and Task Dynamics

In our methodology we combine two distinct dynamic systems to model the movement and behavior of each autonomous agent. The first system controls the movement of the agent. The state space of this system is two dimensional, the first variable represents the heading direction ϕ and while the other specifies forward velocity v. Each autonomous agent's movement is described in polar coordinates. The heading angle is controlled by a one dimensional non-linear dynamical system, which consists of repellers placed in the subtended angle of the obstacles, and attractors in the subtended angles of targets (see section 2.1. In our formulation, the heading speed is modified by the relative location of the obstacles (see section 2.2).

A second system controls the agent's movement decision making, i.e., its behavior. Based on this formulation, an agent ignores targets or obstacles, depending on the scene geometry around the agent at each time instant. This is modeled based on another type of nonlinear dynamical system, running on a faster time scale. This system outputs *weights* that linearly combine the different attractor and repeller contributions as calculated by the first system. The state space of this system is the space of the task constraints. The values of the state vector components determine which "elements" of the environment (e.g., obstacles, targets) will be used in the calculation of the agent's movement and therefore behavior. An important aspect of our methodology is that it scales polynomially with the complexity of the environment.

In the following we present the details of each of each of the two dynamical systems.

2.1. Movement Dynamics I: Heading Direction

The first dynamical system models the control of the basic movement of each autonomous agent. The movement is defined by a 2D vector representing the agent's heading angle and forward speed.

The heading angle ϕ of a given agent is controlled by a dynamical system of the type:

$$\dot{\phi} = f(\mathbf{env}), \tag{1}$$

where **env** is the vector of variables which models the environment (e.g., the geometry and position of the obstacles and targets) and we describe in detail below.

According to our dynamical system formulation each element of the environment can "attract" or "repel" an agent. We will therefore use attractors to model targets and repellers to model objects that should be avoided.

We model an attractor as

$$f_{\text{tar}} = -a \sin(\phi - \psi), \tag{2}$$

where ψ is the angle of the target's location relative to the agent's location and a is a constant parameter.

In order to model complex environment obstacles, enemies, or hazards are distinct entities. Fire-pits, for example, are clearly more dangerous than a large wall. Therefore the repeller definition should have enough parameters to model the different types of objects. We achieve this by defining a repeller to be the multiplication of three different functions, R_i, W_i, D_i, which result in being able to model the type of repeller, its distance to the agent and the extent of its influence to the environment. We therefore repeller as

$$f_{\text{obs}_i} = R_i \, W_i \, D_i. \tag{3}$$

Function R_i models a generic repeller, and is constructed as:

$$R_i = \frac{(\phi - \psi_i)}{\Delta\psi_i} e^{\left(1 - \frac{\phi - \psi_i}{\Delta\psi_i}\right)}, \tag{4}$$

where ψ_i is the angle of obstacle i and $\Delta\psi_i$ is the angle subtended by it.

The second function, W_i, is responsible for limiting the angular range of the repeller's influence in the environment and is modeled as

$$W_i = \frac{1}{2}[\tanh(h_1(\cos(\phi - \psi_i) - \cos(2\Delta\psi_i + \sigma))) + 1], \quad (5)$$

which models a window-shaped function and h_1 is responsible for the inclination of the window's sides and is modeled by

$$h_1 = 4/(\cos(2\Delta\psi) - \cos(2\Delta\psi + \delta)). \quad (6)$$

Here δ is a "safety margin" constant.

The third and last function, D_i, models the influence of the obstacle to the environment by taking into account the distance of the obstacle from the agent and is modeled as

$$D_i = e^{-\frac{r_i}{d_0}}, \quad (7)$$

where r_i is the relative distance between them, and d_0 controls the strength of this influence as the distance changes.

The resulting influence on the agent from all obstacles $i = 1, \ldots, n$, is the sum of the respective repellers

$$f_{\text{obs}} = \sum_{i=1}^{n} f_{\text{obs}_i}. \quad (8)$$

Therefore, the definition of the dynamical system controlling the heading angle in (1) is obtained as:

$$\dot{\phi} = f(\textbf{env}) == |w_{\text{tar}}|f_{\text{tar}} + |w_{\text{obs}}|f_{\text{obs}} + n. \quad (9)$$

The weights w_{tar} and w_{obs} are intended to eliminate spurious attractors that can occur by the direct summing of the nonlinear functions modeling the various obstacles and targets in the environment. These *weights* are obtained through a "constraint competition", the second dynamical system mentioned previously and described in detail in Section 2.3. Finally, the noise term n allows the system to escape from unstable fixed points in the definition of (9) (e.g., the "center" of a repeller, where $\dot{\phi} = 0$, but any slight displacement would make it escape from such a situation such as the situation of a ball situated on the crest of a hill).

The above functions are carefully designed so that certain expected actions will appear in the final system. Let's first consider a simple example (see Fig. 1), the result of a simple interaction between a target, an attractor, and an obstacle, a repeller. Because we have not yet described the constraint competition, let's assume for now that $w_{\text{tar}} = w_{\text{obs}} = 1$. Let's also take the simple case that the location of the obstacle is close to the straight line between the agent and the target.

It is clear that the agent will have to go around the obstacle in order to hit the target. In this case, the modeling of the agent's heading direction $\dot{\phi}$ based on (9) is shown in the lower right graph of Fig. 1. It is the composition of the target function (upper right graphic) and obstacle function (middle right graphic). The presence of two final attractors, indicated by the two arrows in the lower right graph, show the two possible obvious ways to get to the target and avoid the obstacle.

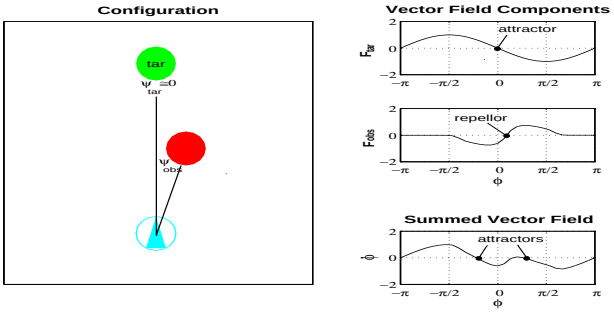


Figure 1. Interaction between one Attractor and one Repeller.

A second more complex example consists of the agent facing two different obstacles located side by side. If the obstacles are too far apart, the agent should be able to pass between them, otherwise it will have to go around them. This decision is taken automatically, as it can be seen in Fig. 2. Fig. 2(a) depicts the case where two obstacles are too close, Fig. 2(b) depicts the case where the distance between the obstacles is exactly equal to the size of the agent, a critical condition, and Fig. 2(c) depicts the case when the obstacles are far apart to allow the easy passage of the agent between them. For this simple case (no target and two obstacles) we have plotted (9) at the bottom of each figure as a function of the angle between the agent orientation and the y axis (assuming that the noise term n is zero). The dark curve is the superposition of the two lighter curves representing the contribution from the two obstacles. These functions clearly show that the dynamical system exhibits the correct behavior in terms of the value of the $\dot{\phi}$. For example in Fig. 2(a) $\dot{\phi} = 0$ depicts an unstable fixed point which would result in the agent trying to go through the obstacles. However, the insertion of a small amount of noise n will overcome this situation easily given the function diagram.

2.2. Movement Dynamics II: Modeling the Agent's Velocity

In the previous section we modeled the change in the agent's heading direction. In this section we model its for-

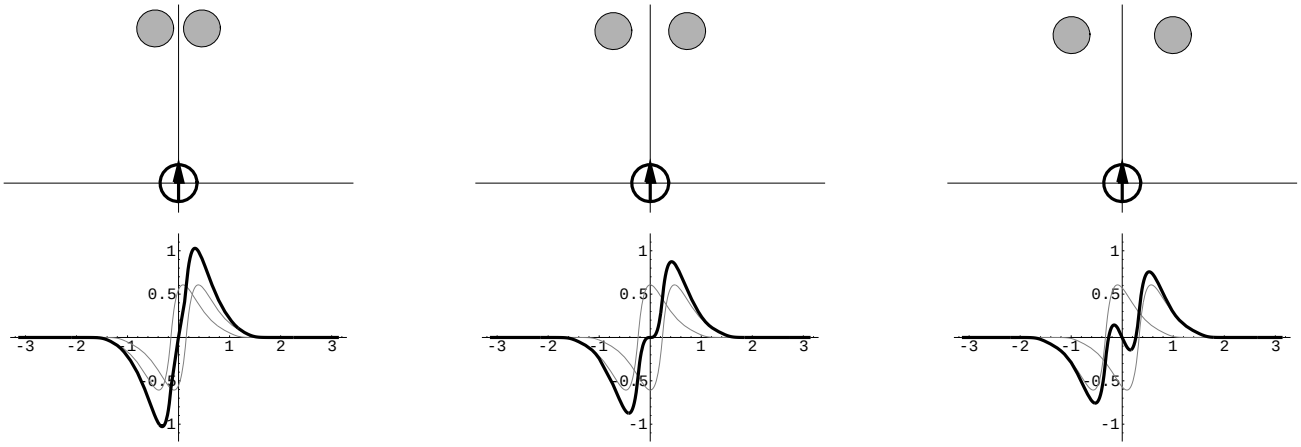


Figure 2. Interaction of two different obstacle repellers.

ward movement. There are a series of different possibilities to accomplish this, but in this paper we discuss three of them. The first is the *constant* velocity, the second is the *time-to-contact* method, and the third is the *derivative* method.

The first option is obviously the easiest one, the agent always moves with a constant speed. This method normally works quite well as long as the agent moves slowly enough. Unfortunately, in some situations it leads to disastrous results. If the heading direction system gets stuck in a spurious attractor, it will most certainly collide since its speed remains constant. Therefore adaptative speeds are necessary in order to achieve a better behavior.

An alternative method seeks to address this problem by making the speed change as a function of the distance between the agent and the closest viewable obstacle. In this time-to-contact method [12] the forward velocity is controlled such that the agent seeks to keep its time-to-contact with upcoming obstacles constant. Using this strategy it is even possible to make a "retreat" (negative forward speed) if this distance is too small.

The idea of the "derivative" method is to slow down when changing direction. A fast change of direction means that either the virtual agent is going in the wrong direction, or even worse, going towards a collision course with an obstacle. In this method, the forward speed is a function of the derivative $\dot{\theta}$ according to the angle (not time).

For the simulations reported in the following sections, forward velocity was controlled using the time-to-contact strategy. However, we continue development of the derivative strategy to provide robust control of velocity in a greater range of situations.

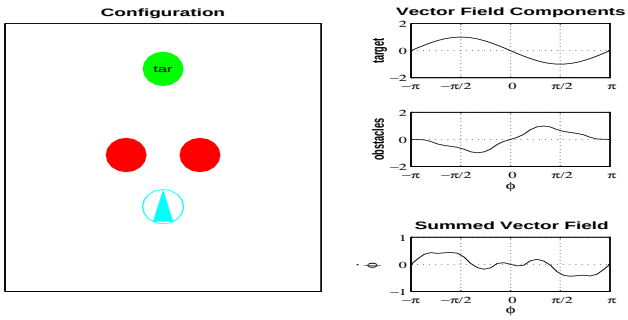


Figure 3. Spurious angle attractor.

2.3. Task Dynamics: Constraint Competition

Individually, the attractors and repellers defined in section 2.1 work well, but because of their non-linear characteristics their direct sum might not always yield the expected results. For instance, in the example shown in Figure 3, the sum of one attractor with the two repellers propose an impossible path in between the two obstacles – they are too close from each other to allow the agent's passage.

To avoid this kind of problem, the composition of the attractors and repeller functions is not obtained by a direct sum, but though an weighted average by weights w_i. These weights are the result of the second dynamical system, which runs at a more refined time scale wrt the dy-

namical system in (9). This second system is modeled as

$$\dot{w}_i = \alpha_i w_i (1 - w_i^2) - \sum_{j \neq i} \gamma_{j,i} w_j^2 w_i, \qquad (10)$$

where in the simple case where only obstacles and targets are modeled, the state space (w_i) consists of ($w_{\text{tar}}, w_{\text{obs}}$), as used in (9).

This system is completely defined by the parameter functions α_i, termed *competitive advantage*, the parameter functions γ_{ji}, termed *competitive interaction*, and the initial value of its state space. At each time instant these parameters will be computed according to the geometry of the environment, and through (10) we obtain the weights to be used in (9). The composite system runs on two time scales – (10) is evaluated at a much faster rate compared to (9). This is in order to ensure that the computed weights result in a stable fixed point of (10) as we will explain below.

The correct design of the parameter functions α_i and γ_{ji} will provide the desired low level behaviors. Therefore it is important to understand the stability of this system (for more details see [15]), and incorporate the geometry of the environment in the "low-level" behaviors. Table 1 shows the stability analysis for (10).

w_{tar}	w_{obs}	Stability	
0	0	Unstable	$\alpha_{\text{tar}}, \alpha_{\text{obs}} > 0$
0	± 1	Stable	$\gamma_{\text{obs,tar}} > \alpha_{\text{tar}}$
± 1	0	Stable	$\gamma_{\text{tar,obs}} > \alpha_{\text{obs}}$
$\pm A_{\text{tar}}$	$\pm A_{\text{obs}}$	Stable and	$\alpha_{\text{obs}} > \gamma_{\text{tar,obs}}$ $\alpha_{\text{tar}} > \gamma_{\text{obs,tar}}$

Table 1. Stability Analysis.

There are four distinct cases each one related to a different behavior. The first case, $(w_{\text{tar}}, w_{\text{obs}}) = (0, 0)$ leads to a situation where the target and the obstacle contributions in (9) are turned off. Obviously this case should be avoided, because the agent would move in an unpredictable way. To avoid this situation, both α_i should always be greater than zero.

The second case $(w_{\text{tar}}, w_{\text{obs}}) = (0, 1)$ occurs when the target's contribution is turned off (like in the case of Fig. 3). It is stable as long as $\gamma_{\text{obs,tar}} > \alpha_{\text{tar}}$.

The third case $(w_{\text{tar}}, w_{\text{obs}}) = (1, 0)$ happens when obstacles are ignored. This may occur, for example, when there are no obstacles near the target. This case is stable when $\gamma_{\text{tar,obs}} > \alpha_{\text{obs}}$.

The last case is when the values of both weights are nonzero, $(w_{\text{tar}}, w_{\text{obs}}) = (A_{\text{tar}}, A_{\text{obs}})$, also known as the "averaging" solution. The following two conditions have to be satisfied for this case to be stable $\alpha_{\text{obs}} > \gamma_{\text{tar,obs}}$ and $\alpha_{\text{tar}} > \gamma_{\text{obs,tar}}$. This is definetely a desirable situation.

It is important to note that conditions two and three are not mutually exclusive, and they can happen simultaneously. In this case we have a situation of *bistability*.

Based on the above, the design of α_i and γ_{ij} should create the different stable points according to the environment parameters. This process is described with details in [10], and the functions for this two-dimensional case are:

$$\gamma_{obs,tar} = \frac{e^{-c_2 P_{tar} P_{obs}}}{e^{c_2}} i, \;\; \gamma_{tar,obs} = 0.05 \qquad (11)$$

$$\alpha_{tar} = a_{tar}, \;\; \alpha_{obs} = \tanh \sum_{i=1}^{n} D_i \qquad (12)$$

where P_{tar} and P_{obs} are:

$$P_{tar} = \text{sgn}(\frac{dF_{tar}}{d\phi}) e^{c_1 |F_{tar}|} \qquad (13)$$

$$P_{obs} = W_{obs} \text{sgn}(\frac{dF_{obs}}{d\phi}) e^{c_1 |F_{obs}|} \qquad (14)$$

and also a_{tar} is such that whenever there is competition among targets and obstacles, targets will loose, but it will always be active if there is only a "background" noise. This is set here to be $0.4(1 - \alpha_{obs})$. D_i (see (7)) is the function used in the distance contribution of each obstacle repeller, and their sum gives a good estimate of the concentration of obstacles around and near the agent.

3. Experimental Results

The system was implemented in C, using *lua* ([17, 16, 1]) as an extensible embedded language to describe both the scene and the target(s)/agent(s) movement.

The constant a in (2) was set to 1 and the safety margin δ in (6) was set to 0.8. The Euler integration time step was 0.25 and all the simulations run in faster than real time.

In the first experiment shown in Fig. 4 we used a single static target and a series of static obstacles between it's location and the target's initial position. Note that in this case d_0 was 3.0.

In the second experiment shown in Fig. 5 the scene is composed of one static target and multiple moving obstacles. The agent avoids collision by changes of direction and sometimes by a velocity reduction or even a complete stop. In this simulation d_0 was set to 2.0.

In the third experiment shown in Fig. 6, there is a group of static obstacles and a moving target. The agent successfully reaches the target and avoids the moving obstacles. In this case d_0 was set to 0.8 and the final velocity was the result of the method scaled by 0.8.

In the last experiment (Fig. 7) we illustrate the flexibility of our method by showing multiple moving and static

targets together with moving and static obstacles . The constant d_0 was set to 1.0.

In the video all the above experiments appear rendered based on the use of the rendering package Pov-Ray [2].

4. Conclusions

We have presented a technique to model autonomous agents for game environments. Using a dynamical system approach we control the agent's heading direction and its velocity. We have demonstrated natural low-level agent behavior in environments with multiple targets and stationary/moving obstacles.

There is a whole set of parameters to control the expected low-level behavior of the overall system. Unfortunately this set is not intuitive for an animator. We are currently working towards making the whole process of modeling a behavior both more high-level and "user-friendly" as well as flexible enough for different applications. New functions are being analyzed in order to achieve a significantly larger and more complex set of behaviors.

5. Acknowledgments

The first author was supported by a Ph.D fellowship from CNPq, Brazil. The third author was supported by a ONR YIP, an NSF Career Award (NSF-9624604), a NASA-96-OLMSA-01-147 and NIST.

References

[1] URL "http://www.tecgraf.puc-rio.br/lua".

[2] URL "http://www.povray.org".

[3] J. Bates, A. Loyall, and W. Reilly. An architecture for action, emotion, and socialbehavior. In *Proceedings of the Fourth Europeans Workshop on Modeling Autonomous Agents in a multi Agents World*, S. Martino al Cimino, Italy, July 1992.

[4] V. Braitenberg. *Vehicles. Experiments in Synthetic Psychology*. MIT Press, 1989.

[5] R. Brooks. Intelligence without reason. Technical Report 1293, Massachussetts Institute of Technology, April 1991.

[6] D. M. E. Kokkevis and N. I. Badler. Autonomous animation and control of four-legged animals. In *Proc. Graphics Interface'95*, Quebec City, Quebec, Canada, May 1995.

[7] R. Grzeszczuk and D. Terzopoulos. Automated learning of Muscle-Actuated locomotion through control abstraction. In *SIGGRAPH 95 Conference Proceedings*, pages 63–70, Aug. 1995.

[8] D. Haumann and R.E.Parent. The behavioral test-bed: Obtaining complex behavior from simple rules. *The Visual Computer*, 4(6):332–347, 1988.

[9] H. Ko, B. D. Reich, W. Becket, and N. I. Badler. Terrain reasoning for human locomotion. In *Proc. Computer Animation '94*. IEEE Computer Society Press, 1994.

[10] C. H. I. Large E. W. and B. R. Scaling the dynamic approach to path planning and control: Competition among behavioral constraints. *International Journal of Robotics Research*.

[11] T. Lethebridge and C. W. C. A simple heuristically-based method for expressive stimulus-response animation. *Computers and Graphics*, 13(3):297–303, 1989.

[12] H. Neven and G. Schöner. Dynamics parametrically controlled by image correlations organize robot navigation. *Biological Cybernetics*, 1996.

[13] H. Noser, O. Renault, D. Thalmann, and N. M. Thalmann. Navigation for digital actors based on synthetic vision, memory and learning. *Computer and Graphics*, 1995. Switzerland.

[14] T. D. Noser H. L-system-based behavioral animation. In *Proc. Pacific Graphics '93*, pages 133–146, 1993.

[15] L. Perko. *Differential Equations and Dynamical Systems*. Number ISBN-0387974431 in Texts in Applied Mathematics. Springer Verlag, Berlin, February 1991.

[16] L. H. d. F. R. Ierusalimschy and W. Celes. *Reference manual of the programming language Lua 3.0*.

[17] L. H. d. F. R. Ierusalimschy and W. Celes. Lua - an extensible extension language. *Software: Practice & Experience*, 26(6):635–652, 1996.

[18] O. Renault, N. M. Thalmann, and D. Thalmann. A vision-based approach to behavioural animation. *The Journal of Visualization and Computer Animation*, 1(1):18–21, 1990.

[19] C. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *Proc. SIGGRAPH '87, Computer Graphics*, volume 21, pages 25–34, 1987.

[20] C. Reynolds. An evolved, vision-based behavioral model of coordinated group motion. In M. Press, editor, *Proc. 2nd International Conf. on Simulation of Adaptive Behavior*, 1993.

[21] G. Ridsdale. Connectionist modelling of skill dynamics. *Journal of Visualization and Computer Animation*, 1(2):66–72, 1990.

[22] G. Schöner and M. Dose. A dynamics systems approach to task level systems integration used to plan and control autonomous vehicle motion. *Robotics and Autonomous Systems*, 10:253–267, October 1992.

[23] G. Schöner, M. Dose, and C. Engels. Dynamics of behaviour: theory and applications for autonomous robot architectures. *Robotics and Autonomous Systems*, 16(2–4):213 – 246, 1996.

[24] A. Steinhage and G. Schöner. The dynamic approach to autonomous robot navigation. In *Proceedings IEEE International Symposium on Industrial Electronics*, 1997.

[25] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of SIGGRAPH '94*, pages 43–50. ACM Press, July 1994.

[26] G. Wilfong. Motion planning in the presence of movableobstacles. In *Proc. 4^{th} ACM Symp. Computational Geometry*, pages 179–288, 1988.

[27] J. Wilhelms. A "notion" for interactive behavioral animation control. *IEEE Computer Graphics and Applications*, 10(3):14–22, 1990.

[28] J. Wilhelms and R. Skinner. An interactive approach to behavioral control. In *Graphics Interface*, 1989.

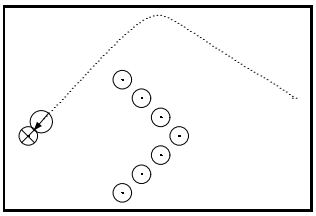
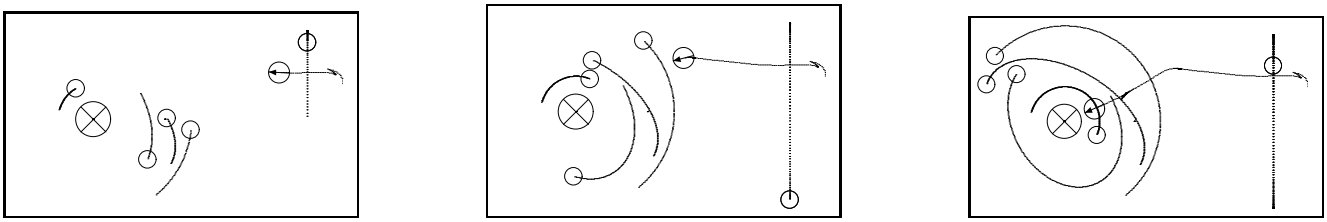
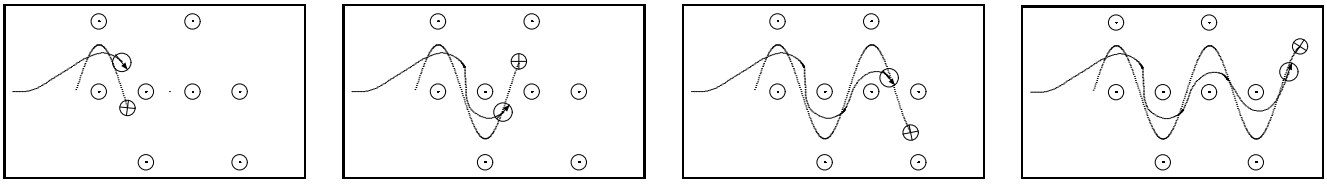


Figure 4. First example: static target and obstacles. The circle with an arrow represents the agent, the circle with the cross is the target and the empty circles are obstacles. Dotted lines show the past trajectory of the moving objects. The same representation applies to the other figures.



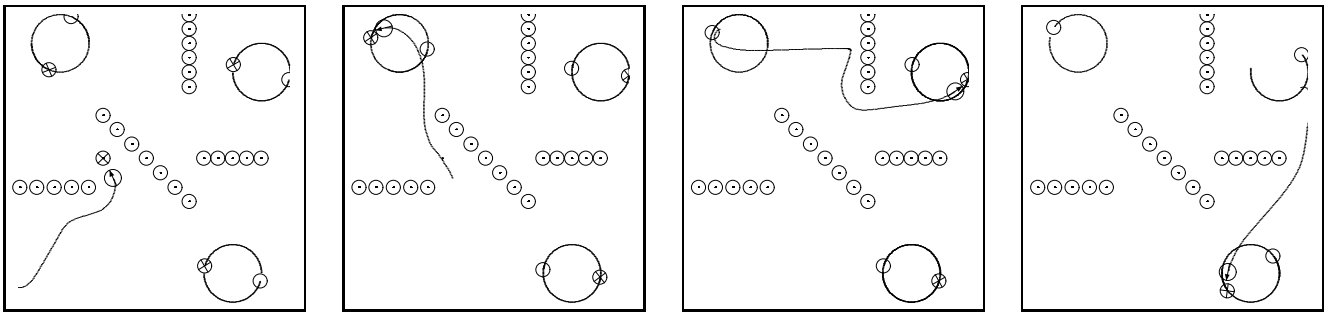
(a) (b) (c)

Figure 5. Second example, static target and moving obstacles.



(a) (b) (c) (d)

Figure 6. Third example, moving target and static obstacles.



(a) (b) (c) (d)

Figure 7. Forth example, multiple moving/static targets and moving/static obstacles.