Social Crowds Using Transactional Analysis

Brian C. Ricks and Parris K. Egbert

Abstract More and more applications are relying on simulated crowds to populate films, games, and architecture. Decades of work in this area has produced agents that deftly avoid collisions, but the crowds still look stiff and false because agents do not socialize naturally with each other. On the other hand, ours is a new, expressive algorithm for adding social dynamics to crowds that breathes a new dimension of realism into simulations. Unlike previous approaches, our work allows agents to have multiple social encounters with other agents. We correctly allow interactions to evolve as time passes using the psychological area of transactional analysis. Additionally, we break from previous paradigms since we do not tie our approach to a specific obstacle avoidance algorithm. Instead our algorithm has a flexible architecture that will run with almost any obstacle avoidance algorithm. Finally, we allow for artist direction in our simulations, including bi-modal crowds and social environments that can be changed in real-time. Our results show that our social crowd algorithm runs in real-time with up to 4,000 agents with far more realistic behaviors than previously simulated.

1 Introduction

Crowd simulation plays a key role in the production of film and games. In the film industry, simulated crowds populate worlds where real crowds would be impossible to create, would be too expensive to be practical, or would not allow for full artist direction. In the game industry, all interactive crowds have to be simulated and more and more games rely on large crowds as part of the plot and gaming experience (see [11]). Even architecture and emergency planning benefit from realistic crowd

Brian C. Ricks
Brigham Young University, Provo, UT 84602 USA, e-mail: bricks@byu.edu

Parris K. Egbert
Brigham Young University, Provo, UT 84602 USA e-mail: egbert@cs.byu.edu

simulation since it allows engineers to anticipate the movement and behavior of people as a building is being designed.

As a result of this real need in industry, the last two decades has seen a strong research interest in crowd simulation algorithms. Unfortunately, most of these algorithms have treated agents more like particles than people since agents usually see other agents only as moving obstacles. This trend runs contrary to psychology research that has shown that up to 70% of pedestrians show social interactions as they walk [7, 15, 19].

Some recent research has recognized the need for social interactions within crowd simulations, but most approaches prove to be insufficient: they do not allow agents to have repeated interactions with different agents, they are built on top of a specific obstacle avoidance algorithm, they are designed only for a very specific social setting, or they do not allow for artist direction. We believe that the power of crowds in film, games, architecture, and emergency planning would all be greatly enhanced with a social crowd algorithm with all of these capacities. Expanding on [30], our algorithm is different from previous work for three specific reasons:

1. Our social crowd algorithm allows social interactions to evolve over time and for agents to have repeated interactions with different agents. In order to achieve this we draw on the area of psychology called transactional analysis that studies such repeated interactions.
2. Our algorithm uses a unique framework for stopping to talk and pair walking behaviors. This framework means our approach is not dependent on a specific obstacle avoidance algorithm.
3. Our algorithm is specifically designed for artist direction. Our results can be easily directed to reflect the different feel of a work place, a public park, a school campus, or a shopping center. Additionally, the feel can be changed in real-time to reflect changes in the environment (such as a when a fire alarm goes off).

In order to validate our algorithm, we demonstrate the existence of all of these features in our results section, Section 7. Our approach runs in real-time with up to 4,000 agents and produces believable crowds, such as those shown in Figure 1.

2 Previous Work

We join previous work in turning to psychology and sociology research to gain insights into improved crowd simulation. Durupinar et al. [8, 9] used the OCEAN personality method [10] to give agents different personality traits. These variations include the speed at which agents move and their response to obstacles (like a preference to move to the right). Pelechano et al. [26] used a presence measure to judge the realism of their crowds. Unlike previous work, we draw on the psychological field of transactional analysis (see Section 3.1).

Fig. 1 Example of a large social crowd on a path at night. Our algorithm uses the area of transactional analysis to create crowds where people move in and out of evolving conversations, stop to talk, and pair walk as shown here.

There have also been many approaches to building social crowds. Musse and Thalmann [20] presented a sociological algorithm that allowed agents to change groups when moving between fixed goals. Their results focused on a museum scenario where such fixed goals would not be unusual. Yeh et al. [33] used proxy agents to model the influence of protection and authority. This was achieved by the creation of spaces that were considered occupied by the crowd simulation engine but which were not rendered as occupied on the screen. Pedica et al. [24, 25] proposed an algorithm for agents talking in groups based on human territory theory. Carstendottir et al. [6] focused on where agents choose to sit in places like a cafe or restaurant.

Traum et al. have done extensive work on dialog for immobile agents. Early research [22, 23] forced agents to be in a constant shared conversation. Later work allowed people to join and leave conversations at will [16] followed by an approach that allowed agents to move to engage different people [17]. This work created believable dialogs but did not deal with large crowds with agents that may move large distances.

Popelová et al. [27] presented research focused purely on two people meeting and then walking together to a destination. Agents wait for each other when needed and walk next to each other using an adapted social forces obstacle avoidance algorithm. They showed that this type of grouping is more believable than the simple leader-follower social setup proposed by Reynolds [28]. Karamouzas and Overmars [18] looked at how small groups of two or three change their formation as they dodge obstacles and other people. Their approach was based on the work of Moussaïd et

al. [19] who did an empirical study of crowds in public places. Moussaïd et al.'s work verified the prevalence of social interaction within crowds.

Such work in social crowds is successful and diverse; however, there are several key dynamics that we feel would make social crowds more believable. First, at work, school, or a public park, people move in and out of different social encounters. Many recent approaches designed for large crowds are not flexible enough to handle both large crowds and the natural evolution of social interactions. Second, most approaches work by manually changing a specific obstacle avoidance algorithm, thus forcing users to the specific advantages and disadvantages of their choice. Third, approaches have focused on a specific social environment (for example people in a museum [20] or people on dates [27]) instead of being designed for artist direction to allow the end user to choose the specific feel of the simulation.

Our social crowd algorithm addresses each of these issues. Unlike previous work, our algorithm allows agents to move in and out of different groups naturally by using transactional analysis. Also, our algorithm has a unique architecture that allows it to run independent of the underlying obstacle avoidance algorithm. Finally, our approach is designed to be artist directed and allows the user to easily change the social environment being simulated.

3 Improving Social Crowds

Fundamentally, crowd simulation algorithms are seeking to optimize a reward function for a group of agents. Traditional crowd simulation algorithms usually try to optimize a function that gives a reward for the speed at which agents reach their destinations while avoiding a large collision penalty. Other variations on this theme include optimizing over agent effort or acceleration (consider [12] and [32]). Although such a speed/collision reward function is simple and crowds designed to optimize such a reward function have some realistic characteristics, this simple approach fails to capture many of the important details of a real crowd.

Observing a real crowd of people shows several dynamics that cannot be explained by a simple speed/collision reward function. For example, when two acquaintances pass they usually stop momentarily to greet and chat. Better friends may spend a significant amount of time talking together without moving towards their destination. Simulating such a commonplace event is impossible with a speed/collision reward function. Similarly, when two friends are walking in the same direction, they often walk right next to each other so they can talk. Again, observation shows that often these "pair walkers" do not follow the shortest path possible to their destination.

More accurate crowd simulations will come with the advent of more expressive reward functions for agents. Instead of balancing only speed to destination and collision avoidance, we believe more accurate crowds will come with an algorithm that balances speed, collisions, and social reward. Looking at to the two previous examples when people deviated from shortest paths in order to stop to talk or pair walk,

these behaviors can be explained by adding a social reward into the overall reward function. For example, people who stop to talk find a greater reward from socializing with each other than they do from continuing on their journey. Similarly, agents who pair walk and thus deviate from their shortest path find a greater reward from talking than from speed alone.

However, not every person stops with every other person in a real crowd. As noted in our introduction, sociological research has shown that up to 70% of agents show socializing behavior in real crowds at some point, but not everyone is socializing at every moment in a crowd. Again, this can be explained using our proposed reward function. In cases when people do not stop to talk or pair walk, their reward from socializing is less than the reward of reaching their destinations quickly.

It follows that a crowd simulation algorithm that allows for socializing needs some method of determining whether a given agent will find more reward from socializing or from hurrying directly towards its destination. We can do this using a function with with the following signature:

$$SocialReward : a, b \rightarrow reward \tag{1}$$

where a as the agent in question, b is a nearby agent with whom a could socialize, and *reward* is a scalar that indicates the reward for a and b when they socialize. If *reward* is greater than the reward from reaching the current destination as fast as possible, then we would expect a to socialize with b.

One common approach to calculating *SocialReward* is to return one of two values: a high value if a and b are friends and 0 if they are not. Both Popelovà et al.'s work [27] and Karamouzas and Overmar's work [18] can be thought of as taking this approach. Unfortunately, this all or nothing method fails to create the realistic social dynamics seen in real crowds, as discussed previously. For example, if we use this method and agents a and b approach each other in opposite directions, they would stop and talk but to never stop talking. If the *SocialReward* function always returns a high value, the agents would never find it more rewarding to move on than to continue talking, and the crowd would quickly degenerate into pairs of friends talking indefinitely. Similarly, if a and b were heading in the same direction, they would pair walk but never break away from each other to reach their respective destinations.

As noted by James and Coleman [7, 15] in studies on real crowds of people, each person usually moves in and out of different conversations with different people as they move to their destination. We follow this line of reasoning and propose that the most realistic social crowds will come when *SocialReward* returns more than one of two values and instead accounts for changes in social interest based on the passing of time and past history.

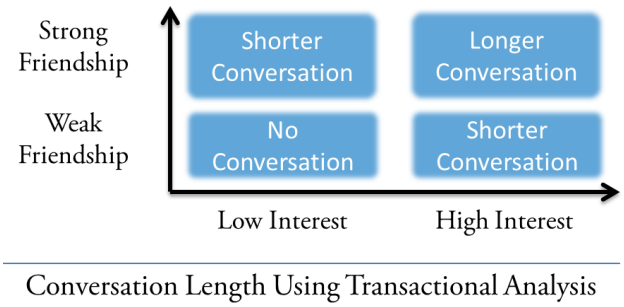


Fig. 2 Graph showing how friendship and interest change the type of conversations each agent has. If the friendship is higher, the conversations will be longer than if the friendship is weaker. Similarly, if the agents' interest is higher because they have not talked recently the conversation will be longer than if the interest is low because they recently saw each other.

3.1 Transactional Analysis

We are aided by the fact that for decades sociologists and psychologists have studied how interest in conversation evolves over time. In order to create a more realistic *SocialReward* function, we turn to this research, specifically the area of transactional analysis.

Introduced by Eric Berne in the 1950s [2], transactional analysis has studied the how and why of conversation in a rigorous way. By now research in transactional analysis has grown to include its own scholarly journal and professional association. The research has produced countless scholarly articles and a myriad of popular psychology books including [3, 4, 5, 13]. Most relevant to our research is transactional analysis' look at the expected length of a spontaneous social interaction or "rituals." The length of the conversation is often measured in terms of social "strokes," or moments when each person gives the other person attention.

Transaction analysis asserts that the length of a ritual, or the amount of expected strokes in an interaction, is dependent on two main variables: the relationship between the agents involved and the history of their recent social interactions [4]. Again, looking at real people in real crowds, one would expect good friends to stop and talk for a long time while such a lengthy engagement would seem awkward or culturally unexpected between strangers. Similarly, one would not expect two people to have a lengthy conversation if they have recently had another lengthy conversation. This can frequently be observed in an office environment. The first time an office worker passes a secretary it would not be unusual for that person to stop and talk for a while. If that same office worker walked back past the secretary a few moments later, we would not expect more than a short greeting. If however, that person did not come by for a long time or until the next day, we would expect to see a return to a more protracted conversation. These dynamics are shown in Figure 2, which shows the expected conversation length based on friendship and recent history.

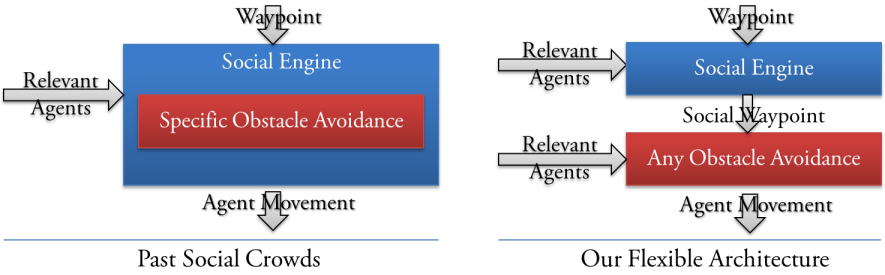


Fig. 3 Examples of traditional social crowds architecture on the left and our flexible architecture on the right. Notice that our approach does not rely on a specific obstacle avoidance algorithm but instead creates temporary social waypoints for each agent.

In order to capture these dynamics as outlined by the transactional analysis literature, our work implements *SocialReward* differently than previous work. Denoting our function $SocialReward_{ta}$ to indicate that we are using transactional analysis principles, our approach can be formalized as follows:

$$SocialReward_{ta} : a, b, History(a,b) \rightarrow reward \tag{2}$$

The difference between *SocialReward* and $SocialReward_{ta}$ is the introduction of a new argument, $History(a,b)$ which represents the past history of a and b's interactions. Using $SocialReward_{ta}$ we can create agents that stop to talk and pair walk based not only on the strength of their relationship but also on their recent interactions. We detail our implementation of $SocialReward_{ta}$ in Section 4.

3.2 New Architecture

Before we detail our implementation of $SocialReward_{ta}$, we address another significant difference between our work and previous approaches. In addition to determining which interactions are rewarding, a social crowd algorithm needs to maneuver agents in and out of rewarding conversations. Turning again to sociology and psychology, with every conversation there is a mutually understood center called the formation nucleus (consider Scheflen and Ashcraft [31] or Pedica and Vilhjàlmsson [25]). A social crowd simulation algorithm needs to move agents towards their nucleus until they are close enough to engage in conversation. If the agents are pair walking this is even more complicated since the formation nucleus will move as the agents move.

In order to move agents in and out of range of a formation nucleus, almost all previous work has taken the approach of manually altering a specific underlying obstacle avoidance algorithm. For example, if an algorithm is using social forces like [27], the social algorithm would alter the implementation of social forces to allow agents to move toward a given formation nucleus. The same could be said for

other popular algorithms like velocity-based approaches (as seen in [18]). However, this implementation choice means that each social crowd algorithm is permanently tied to a specific obstacle avoidance algorithm (consider Figure 3).

We believe that a far more useful architecture will decouple the choice of obstacle avoidance algorithm and the social engine, as shown in Figure 3. To do this, we allow our social engine to temporarily alter each agent's destination based on social factors, but not to make any changes to the obstacle avoidance algorithm. This approach, which we detail further in Section 5, means that individual users of our social crowd algorithm are not tied to a specific obstacle avoidance algorithm.

4 Implementing Transactional Analysis

In the previous section we discussed the theory of social crowd simulation. In this and the following two sections we give the details of our implementation starting with transactional analysis.

As discussed in Section 3.1, transactional analysis presents a scientific way of predicting the amount of time that a conversation will be rewarding to participants. This length is calculated using two variables: the relationship of the agents and their recent history of conversation. In order to add this to our social algorithm, we quickly need to figure out the relationship between two agents and a simple way to store the relevant information of their past interactions.

4.1 Determining Relationships

Returning again to real life relationships, we note that most relationships are reflexive but not necessarily transitive. Relationships are generally reflexive since they are built through mutual interaction. Formally, if A is the set of all agents in our simulation, then we want a relationship model where $\forall a,b \in A : \mathit{SocialReward}(a,b) = \mathit{SocialReward}(b,a)$. However, even if relationships are reflexive, we do not necessarily assume that they are transitive. Specifically, if $\forall a,b,c \in A : \mathit{SocialReward}(a,b) = \mathit{SocialReward}(b,c) \not\Rightarrow \mathit{SocialReward}(a,b) = \mathit{SocialReward}(a,c)$. All the same, we would expect the probability of $\mathit{SocialReward}(a,c)$ being related to the values of $\mathit{SocialReward}(a,b)$ and $\mathit{SocialReward}(b,c)$ since friends of friends are more likely to be friends.

We have found that a simple way to create a set of agents with these properties of reflexivity and intransitivity is to give each agent a friendship base score between 0 and 2π. The strength of the relationship between any two agents a and b can be found by taking the angle difference between the friendship base score of a and b. If the angle difference is low, the relationship between them is strong and they will be highly rewarded by talking. If the distance is high, they do not have a strong relationship and will not find a lengthy conversation rewarding.

There are a variety of ways to calculate angle difference. For our work the following formula worked successfully:

$$AngleDifference(a,b) = \begin{cases} a-b & if\ (-\pi < a-b < \pi) \\ a-b+2\pi & if\ a-b \leq -\pi \\ a-b-2\pi & if\ \pi \leq a-b \end{cases} \quad (3)$$

Another way of thinking about this is that a and b are located on a unit circle at the angle corresponding to their friendship base score. If the distance between them on the outside of the circle is short, they are good friends. If the distance is long, they are not good friends. It follows that using this method creates relationships that are reflexive since the distance between two angles is reflexive. It also follows that the relationships are not necessarily transitive since angle differences are not transitive. Notice also that if a and b have a low angle difference and b and c have a low angle difference, then a and c are likely to have a low angle difference. Thus, this simple approach gives us all the desired friendship properties.

In Section 6 we discuss how we distribute the friendship base scores of our agents to allow for artist direction in the creation of a scene.

4.2 Storing Past History

The transaction analysis literature suggests that people find a diminishing reward from socializing when they have socialized recently. Formally, we call this the interest of two agents interacting I_{ab} if a and b are the agents in question. Since agents have different interest levels for each other agent, agents store an array of scalars that indicate their interest level relative to each other agent. We discuss optimization to this approach later.

In our section on artist direction, Section 6, we discuss how global variables can change this interest level, but for now we assume these global values are set to identity. At the beginning of the simulation we initialize all interest levels to 1, i.e., $\forall a,b \in A, a \neq b, I_{ab} = 1$. The question therefore becomes how to evolve the interest between agents to reflect social interaction, or lack thereof, as the simulation moves from the current timestep t to the next timestep $t+1$.

In order to calculate changes in interest we introduce two simple functions: a *Near* function and a *Socializing* function. $Near(a,b,t)$ is true iff a and b are close enough to be socializing at timestep t, and $Socializing(a,b,t)$ is true iff a and b are actually socializing at timestep t. It follows that $Socializing(a,b,t) \Rightarrow Near(a,b,t)$. Note than $Near(a,b,t)$ is easy to calculate since crowds simulations almost always use a grid structure to speed up relevancy calculations (see [29] for further discussion). $Socializing(a,b,t)$ involves a simple lookup and is likewise computationally inexpensive.

The first step in evolving interest levels between agents is to increase their interest if they are not socializing. Since $\neg Near(a,b,t) \Rightarrow \neg Socializing(a,b,t)$, if a and b are not near each other at timestep t then we increase I_{ab} at timestep $t+1$ as follows:

$$I_{ab,t+1} = I_{ab,t} + .01 \; if \; \neg Near(a,b,t) \tag{4}$$

The second step involves decreasing interest if agents are socializing as follows:

$$I_{ab,t+1} = I_{ab,t} - .03 \; if \; Socializing(a,b,t) \tag{5}$$

In both of these steps our choice of amount to increase and decrease interest is arbitrary. However, we have found in practice that for realistic results interest should increase slower than it decreases when agents are socializing. Further, we have found that two unnatural results can occur if additional steps are not added to our calculations. First, agents can be near each other but not socializing. For example, agent a and b may be socializing and c and d may be socializing with both of these groups being very near each other. If a,b,c, and d are all near each other, an unnatural conversation oscillation can occur. If a and b end their conversation about the same time that c and d end their conversation, then a and c and b and d might immediately begin socializing. When their conversations end, a and b might return to talking and c and d might return to talking. This can go on forever and does not create believable results. We therefore add the following calculation:

$$I_{ab,t+1} = I_{ab,t} - .01 \; if \; Near(a,b,t) \wedge \neg Socializing(a,b,t) \tag{6}$$

This calculation implies that agents that are near each other but not socializing exchange some non-verbal communication (or a simple hello) that results in a slight decline in interest.

We also enforce a dramatic decrease in interest when a conversation ends to prevent other conversation oscillations as follows:

$$I_{ab,t+1} = I_{ab,t} - .5 \; if \; Socializing(a,b,t-1) \wedge \neg Socializing(a,b,t) \tag{7}$$

Combined these four calculations can be written as:

$$I_{ab,t+1} = \begin{cases} I_{ab,t} + .01 \; if \; \neg Near(a,b,t) \\ I_{ab,t} - .01 \; if \; Near(a,b,t) \wedge \neg S(a,b,t) \\ I_{ab,t} - .03 \; if \; S(a,b,t) \\ I_{ab,t} - .5 \quad if \; S(a,b,t-1) \wedge \neg S(a,b,t) \end{cases} \tag{8}$$

We also prevent interest levels from growing above 1 since there is a maximum interest that two parties will have. With this cap, we can store interest levels sparsely for each agent. Agent a will only store interest values for another agent $b \in A$ if $I_{ab,t} < 1$. In large simulations (like ours with 4,000 or more agents) the majority of interest levels will be at 1. This means our optimization provides a space and time savings.

As we discuss in our results section, implementing transactional analysis in this way results in agents who have multiple realistic conversations with different agents over time that match what we see in real life.

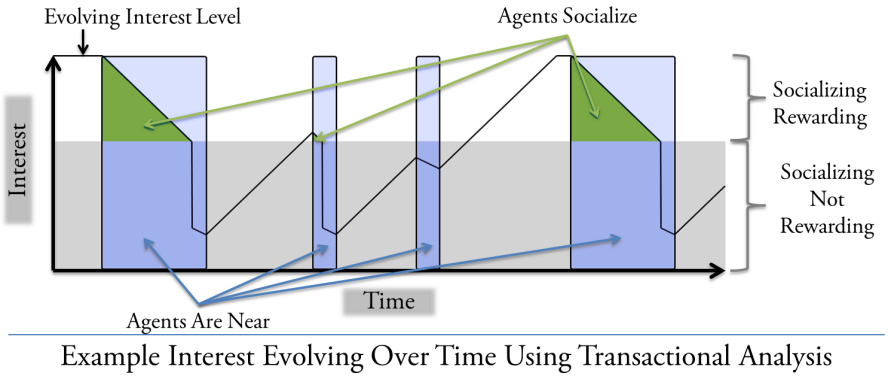


Fig. 4 Graph showing how an agent's interest in talking with another agent evolves over time in a scenario like that shown in Figure 8. The vertical axis shows interest. Interest levels high enough to produce conversation are highlighted white and interest levels too low to produce conversation are shaded in gray. The horizontal axis shows time. Areas shaded blue are times when the agents are close enough to socialize and areas shaded green are times when the agents actually engage in conversation. Notice that in the first encounter the agents socialize for a long time until their interest drops below the threshold, at which point it drops dramatically (see Equation 8). Upon meeting again, the agents have a short conversation and later they don't have any conversation since their interest is too low. When they meet again much later, the interest has risen and they again have a long conversation.

4.3 Individual Mood

In addition to giving each agent a friendship base score and interest values, we give each agent a mood. The mood for each agent a, M_a, is a single scalar that reflects agent a's interest in socializing. A mood of 1 indicates a high interest in socializing and a mood of 0 indicates no interest in socializing. The difference between agents with high and low mood values are easily seen in our simulation: Agents with higher values stop to talk with more people and their conversations are longer, while agents with lower values talk with fewer people and for less time. Use of a mood value again gives us further artist direction capacities in designing a social environment.

4.4 Social Reward

Using our friendship base angle, interest levels, and agent mood as just described, we can now give our implementation of $SocialReward_{ta}$ as originally discussed in Section 3.1.

$$SocialReward_{ta} = \\ AngleDifference(a,b) \cdot min(I_{ab,t}, I_{ba,t}) \cdot min(M_a, M_b) \cdot GlobalInterest \quad (9)$$

We use a *min* calculation in the event that $I_{ab,t} \neq I_{ba,t}$. This should not happen in our current simulation, but we are looking at future work where different types of agents may have different personalities and as a result different rates of change in their interest. For our implementation, an agent will socialize if *SocialReward* is greater than .5. If not, the agents will head toward their respective destinations. We discuss *GlobalInterest* further in Section 6 since it is part of our artist directed capabilities.

5 Implementing Flexible Architecture

As noted in our theory section, Section 3.2, the second main contribution of our social crowd algorithm is a flexible architecture for moving agents in and out of a formation nucleus when socializing. Specifically, we do not tie our implementation to a specific obstacle avoidance algorithm. Instead, we temporarily alter each agent's waypoint to reflect the current social situation.

We allow for two main types of agent interaction. First, we allow agents to stop and talk when they are heading in opposite directions. Second, we allow agents to pair walk, or walk in tandem, as they head toward their destinations. Previous work generally focuses on one or the other of these phenomenon, but we provide an implementation that allows for either of these two interactions based on the social circumstance.

5.1 Stopping and Talking

Stopping to talk happens when two agents recognize that they will have a rewarding conversation and also recognize that their destinations are in different directions. In our implementation, agents will want to stop to talk as opposed to pair walking when the cosine of the their destination angle difference is less than 0.

Stopping to talk has two distinct phases. In the first, agents walk towards each other until they are within their formation nucleus. In the second phase the agents stop moving and talk until the conversation ceases to be rewarding. In assigning temporary social waypoints to agents who are stopping to talk, we account for both of these phases. If agents are not within their formation nucleus, then their temporary social waypoint will be in the direction of the other agent. If the agents are within the formation nucleus, the temporary social waypoint is simply their current location.

Formally, the temporary social waypoint for a when stopping to talk to b is as follows:

$$StoppedWaypoint(a,b) = \begin{cases} a + ||b-a|| & if\ |a-b| > 1.5m \\ a & otherwise \end{cases} \tag{10}$$

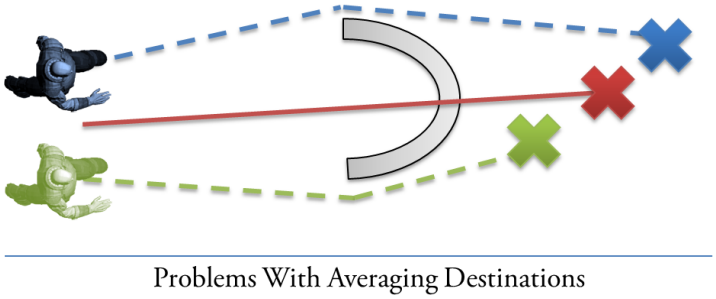


Fig. 5 Simply using the average of pair walking destinations can lead agents into local minima, as shown here. The blue (darker) agent's destination is denoted by the blue X, the green (lighter) agent's destination is denoted by the green X, and the average is denoted by the red X in the middle. If our algorithm assigns a temporary waypoint at the average of the blue and green agents' destinations then they will get stuck in the concave wall. Instead we use a dominant agent strategy to remove the agents stalling as they pair walk.

5.2 Pair Walking

Pair walking occurs when two agents walk together because their destinations are in roughly the same direction. Since this means that the formation nucleus of the conversation is moving, this can prove to be a significantly more challenging problem. However, we have a straightforward and efficient method for reproducing this behavior in a crowd.

Agents start pair walking if the cosine of their angle difference is greater than 0. Notice that this does not imply that the destinations are in the exact same location, which we take into account later. Once agents begin pair walking, we need to figure out how the formation nucleus will move as the agents move. One failed approach is to take the average of the agents destinations and move the formation nucleus in that direction. This seems like a plausible approach since it does not sacrifice the speed reward of one agent in favor of the other. However, in practice, moving agents in the average direction of their destinations can lead them into obstacles. If an obstacle is concave, the agents can end up in a local minimum where they cannot escape without replanning their paths (consider Figure 5).

To remove this unrealistic behavior, we instead move the formation nucleus along one of the two agents' paths. To do this we choose a "dominant" agent by taking the one with the lower internal ID number and the formation nucleus remains near this dominant agent. In the future we are looking to more psychologically-based approaches for choosing dominant agents.

In order to move the formation nucleus using only temporary social waypoints, we again recognize two distinct cases: times when agents are within the formation nucleus and times when some obstacle has split the agents and they are no longer within the nucleus. In the case where the agents are within the formation nucleus, the agents continue walking at a comfortable pace until their real destinations diverge.

Fig. 6 How a pair walking agent adjusts its speed based on the angle to the other agent. If the other agent is in front of the agent in question (angles highlighted green), the agent in question moves at full speed. If the other agent is more than a little behind (angles highlight orange) the agent in question slows proportionally. If the other agent is almost directly behind the agent in question (angles highlighted bright red) the agent in question approaches a full stop. Using this approach agents are usually able to stay abreast and if they are separated they start and stop naturally.

In the case where the agents are not within the formation nucleus, we slow down the agent who is in front until the other one catches up (see Figure 6).

We could have agents turn around to meet back up again, but as noted in Popelová et al. [27], stopping to wait generally looks more believable. Additionally, we have found that dramatic changes in speed, for example if the agent in front stops suddenly, look less natural than slowing down as agents slip out of the formation nucleus. To simulate these natural changes in speed, we throttle the speed of the agent in front based on the cosine of the angle to the front agent's destination and the angle to the lagging agent. If the cosine is greater than $-.1$ than the agents will walk at full speed. Otherwise the agent in font will slow down proportional to the cosine. The lower the cosine, the slower the agent in front will go. In practice we have found that this creates very natural pair walking even in environments where there are lots of obstacles and agents are frequently separated.

Centering the formation nucleus directly above the dominant agent can also create small but regular oscillations in agent movement. This happens as the less dominant agent is drawn toward the formation nucleus then repelled by the obstacle avoidance algorithm to prevent a collision. As the less dominant agent angles away, the obstacle avoidance algorithm may no longer recognize the impending collision and the agent will turn back, and so forth, for the length of the journey. To prevent these oscillations, we offset the formation nucleus from the dominant agent to the left or right based on the location of the less dominant agent (see Figure 7).

To offset the formation nucleus (and the corresponding temporary social waypoint) our algorithm finds the line from the dominant agent to the dominant agent's

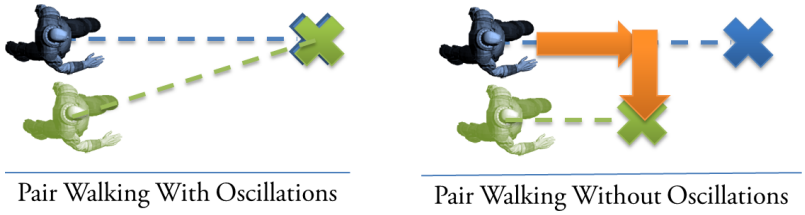


Fig. 7 Minor oscillations are visible when agents pair walk without separating their temporary social waypoints. On the left two agents without this improvement head to the same point and will begin oscillating as the obstacle avoidance algorithm tries to prevent a collision. On the right our algorithm assigns a temporary social waypoint that is a shoulder's width away from the blue (darker) dominant agent for the less dominant green (lighter) agent.

waypoint. Then we use simple geometry to determine on which side of that line the less dominant agent's location falls. The less dominant agent's temporary social waypoint is then offset in that direction by a shoulder's width. This removes oscillations and agents move naturally as a pair until their destinations diverge.

Calculating temporary social waypoints for stopping to talk and pair walking produces very believable crowds without tying our implementation to a specific obstacle avoidance algorithm. As discussed in our results section, Section 7, we used three different obstacle avoidance algorithms in our implementation, each of which produced realistic social crowds without manually altering the obstacle avoidance algorithms.

6 Implementing Art Direction

Many of the papers noted in our previous work section, Section 2, proposed social algorithms for specific social environments. For example, Musse and Thalmann [20] focused on a museum environment, Popelová et al. [27] focused on people who are permanently paired off, and Karamouzas and Overmars [18] looked only at small groups like shoppers at a mall. These approaches are successful in reaching their goals, but none of them are expressive enough to create a large range of social environments nor are they flexible in letting the user easily choose the specific environment.

Our approach is unique when compared to previous work because our algorithm is expressive in the types of social crowds it can simulate. Notice that in our section about transactional analysis, Section 4, we gave two values to each agent that determine its social behavior—a friendship base score and a mood. By altering the distribution of these two values across all the agents in our simulation we can create a huge range of social environments, including many of those proposed in previous work in addition to new ones. Examples of these different environments and the corresponding distribution of relationship values and mood values are shown in Table 1. For example, we can create an office scenario where most people chat briefly

Simple Environments

Scenario	Desired Effect	Relationships	Mood
Office	Most agents stop to talk briefly with others	Lots of friends	Just enough for brief chats
School	Agents talk longer, lower percentage of friends	Fewer friends	Higher
Park	Some people are friends	Medium friends	Medium interest

Bi-Modal Environments

Scenario	Desired Effect	Relationship	Mood
Party Participants	Agents talk for a long time, but not with waiters	High with all party goers	Higher
Party Waiters	Agents move without little or no interaction	None with party goers	Low

Real-time Changing Environments

Scenario	Desired Effect	Global Interest
Panic	Agents stops talking and hurry to exits	Drop to low
Work Day	People are more willing to talk as they start the day and later in the afternoon	Higher early and late, low in middle
High School	People talk during breaks then rush to class	High during breaks, low when bell rings

Examples of Artist Directed Options in Our Algorithm

Table 1 Different social scenarios we can create using the artist directed nature of our scene.

as they pass or a school environment where people have proportionally less friends but chat longer by simply altering these distributions.

The most interesting and unique environments are created when we allow for either a bi-modal distribution or real-time changes in the distribution. Again, as shown in Table 1, if we give half the agents one distribution and the other half another distribution, we can create convincing scenes sucha as people at a dinner party. If the ”party goer” agents are given higher mood values and have close friendship base scores to each other, they will chat at length with most of the other party goers. By giving the waiter agents lower mood values, they move about without stopping to talk as they do their work.

Similarly, changing the distribution over time can result in other striking simulation results. For example, if we reduce the interest value of all agents simultaneously, we can easily recreate a fire alarm scenario. Agents talk like normal until a button is pushed that drops the interest level. The agents then move directly to their destinations. Upon pushing the all clear button the user can bring the interest levels back up and the agents resume socializing naturally. We give snapshots of such a scenario in Figure 10. In order to facilitate global changes in interest, we found it easiest to add a *GlobalInterest* value to our calculations, as shown in Equation 9. By attaching the value of *GlobalInterest* to the state of the user-controlled fire alarm button, we add real-time artist direction to our algorithm. Other time-dependent changes include the natural change of mood throughout the day in a workplace or the change in socializing at school before and after a bell rings.

Combined, the ability to alter the distribution of agent values and change the global interest in real-time makes our approach a unique and powerful artist directed social crowd simulation.

7 Results

Our algorithm is different than previous social crowd algorithms for three reasons: our algorithm uses transactional analysis to move agents in and out of multiple interactions, our algorithm is built using a flexible architecture, and our algorithm allows for artist direction during the simulation. To validate these contributions, we ran our algorithm using a variety of simulations (for example, see Figure 1 and Figures 8-11) and looked for the presence of each of our contributions. We detail the results of all three of these areas now, followed by a look at the performance of our algorithm.

7.1 Transactional Analysis Results

We verified our social crowd algorithm by looking for agents moving naturally in and out of multiple conversations using the principles outlined in the transactional analysis literature. In each of the simulations we ran, agents had multiple conversations, the length of which evolved based on past interactions and reflected the friendship base score and mood of each agent.

One example that verifies our implementation is shown in Figure 8. This figure shows two agents who have four encounters over time. In the first encounter the agents talk at length (noted by word bubbles above each agent). In the second encounter the agents still stop to talk but chat only briefly. In the third encounter the agents do not talk since they have recently talked twice already. In the fourth encounter, which happens much later, their interest in talking has returned and they again stop for a lengthy conversation. Notice that this encounter follows the pattern

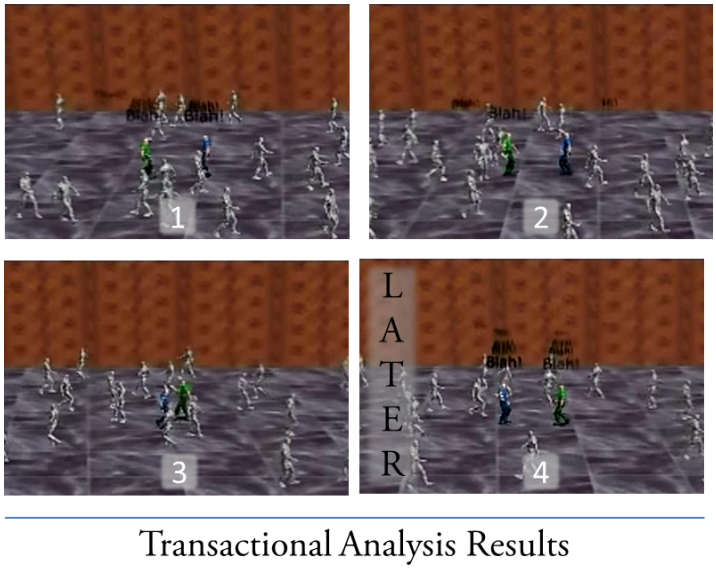


Fig. 8 Results of our algorithm based on transactional analysis. As the same agents meet repeatedly, the conversation length, denoted by word bubbles, shortens until they do not stop to talk when they meet (frame 3). After a long time passes (frame 4), the agents again engage in conversation.

established by transactional analysis for repeated social interaction (see Section 3.1 and Section 4). Figure 4 shows a graph of interest from a very similar scenario.

7.2 Flexible Implementation Results

Our second contribution centers around a flexible architecture that does not modify a specific obstacle avoidance algorithms. This contribution is straightforward to demonstrate because we ran our algorithm using three very different underlying obstacle avoidance algorithm. We implemented Helbing and Molnar's social forces [14], reciprocal velocity obstacles based on van den Berg et al. [1], and the anticipation model proposed in Ondřej et al.'s work [21] and used each of these with our social algorithm.

With every obstacle avoidance algorithm, our crowds displayed stopping to talk and pair walking behavior' as described in Section 5. An example of stopping to talk can be seen in Figure 8. Figure 9 shows an example of two agents pair walking. In the first frame they wait for each other and start talking. In the second frame they walk toward the dominant agent's destination. In the third frame the angle to their destinations has diverged and they part ways.

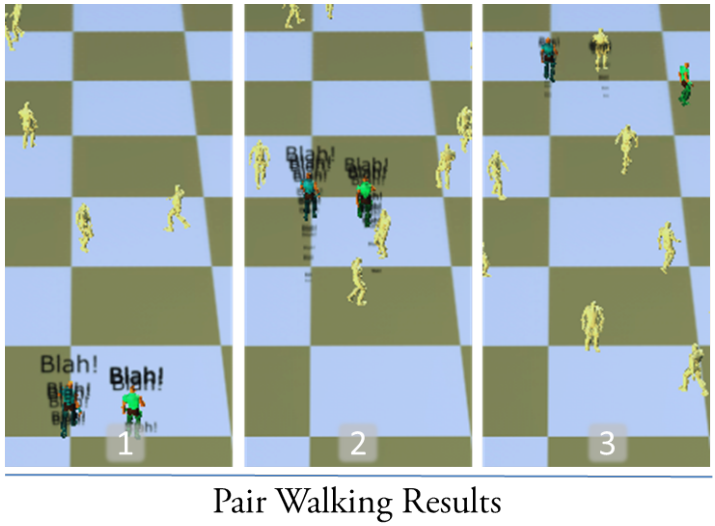
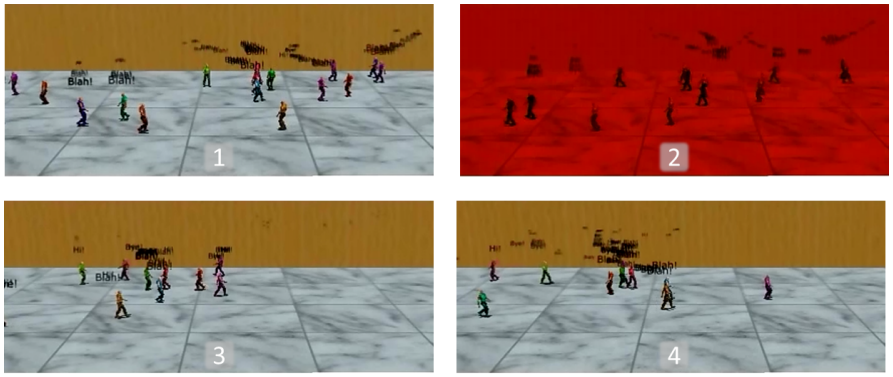


Fig. 9 Example of pair walking using our flexible architecture. In frame 1 the agents meet, then walk together (frame 2) until their destinations diverge. In this case they part ways and head toward their real destinations (frame 3).

7.3 Artistic Direction Results

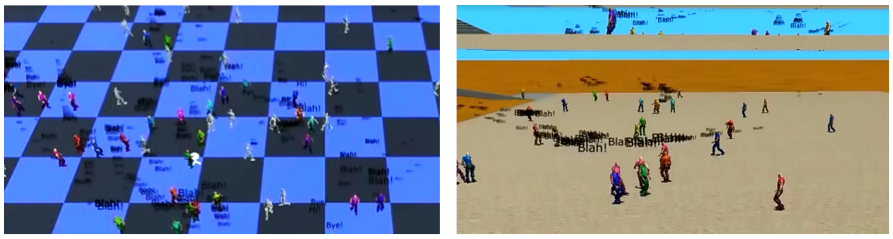
The last contribution of our work is easy artist direction. To validate that our algorithm allowed for such artist direction, we added simple buttons to our user interface that allowed the user to change the distribution of friendship base scores and mood values among the agents. We found that we could easily create a variety of different social environments like those listed in Table 1 using our interface.

A specific example of such artist directed control is shown in Figure 10. In this scenario agents are walking around at work. The first frame shows agents conversing as the user pushes the fire alarm button. This button reduced the value of *GlobalInterest* (see Equation 9) to 0 and gave each agent a destination at the nearest exit. Notice in the second frame that the conversation has stopped and agents move quickly to exits. The third frame shows agents reengaging in conversation when the user pushes the all clear button that raises *GlobalInterest* to 1 and gives agents normal destinations again. The fourth frame shows the agents walking normally now that the fire alarm is over. Further artist direction results can be seen in Figure 11 which shows on the left results from our bi-modal party distribution as explained in Table 1.



Fire Alarm Results

Fig. 10 Example of our real-time artist direction in a fire alarm scene. In frame 1, the fire alarm goes off so the global conversation interest drops and socializing stops (frame 2). In frame 3, the user gives the all clear and agents return to conversations and soon regain their previous social dynamics (frame 4).



Bi-Modal Party Results

Multi-story Office Scenario

Fig. 11 Two more examples of results with our algorithm. On the left we create a party scene (waiters in gray) using a bi-modal distribution as described in Table 1. On the right we show our algorithm running in a multi-story building scenario.

7.4 Performance Results

We further validated the contribution of our approach by looking at its performance. As shown in Table 2, we ran our algorithm using 1,000 to 4,000 agents and calculated the frame rate. Even with 4,000 agents our algorithm ran at real-time speeds. Our tests were done on an Intel i-7 2600 chip at 3.4Ghz with our program consuming only 10-15 percent of the CPU time. Profiling our program demonstrated that the code we added for transactional analysis and our new architecture consumed less than 3% of our computation time in our scenario with 1,000 agents. This minimal impact suggests that our approach could be easily added to existing crowd simulation algorithms without a meaningful increase in computation time.

Agents	Frames Per Second	Agents	Frames Per Second
1,000	57.2	2,000	53.7
3,000	30.0	4,000	23.5

Performance Results

Table 2 Performance results with our algorithm. Notice that even with 4,000 agents our algorithm ran in real-time with all our social interactions.

8 Future Work

In terms of future work, we are looking at several further improvements to our algorithm. First, our algorithm does not easily allow for the creation of groups larger than two people. Sociological work has found that the equilibrium nature of crowds tends toward groups of one or two, with less than 12% of observations showing groups of size three or more (see James [15] and Coleman [7]), but we are still interested in adding this capacity to our approach. Second, we are interested in more expressive mood models for our agents based on psychological research. For example, we might have even more realistic crowds if we use the OCEAN personality model [10] in our work. Third, we are looking at more psychologically-based approaches to choosing dominant agents. Lastly, we are looking into adding motion capture data into our social crowds so that as agents talk they have visible forms of non-verbal communication.

References

1. Van den Berg, J., Lin, M., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on pp. 1928–1935 (2008)
2. Berne, E.: Transactional analysis: a new and effective method of group therapy. American Journal of Psychotherapy **12**(4), 735 (1958)
3. Berne, E.: What do you say after you say hello?: The psychology of human destiny. Bantam (1984)
4. Berne, E.: Games people play: The psychology of human relationships. Penguin (2010)
5. Berne, E., Steiner, C., Kerr, C.: Beyond games and scripts. Ballantine Books (1981)
6. Carstensdottir, E., Gudmundsdottir, K., Valgardsson, G., Vilhjalmsson, H.: Where to sit? the study and implementation of seat selection in public places. Intelligent Virtual Agents pp. 48–54 (2011)
7. Coleman, J., James, J.: The equilibrium size distribution of freely-forming groups. Sociometry **24**(1), 36–45 (1961)
8. Durupinar, F., Allbeck, J., Pelechano, N., Badler, N.: Creating crowd variation with the ocean personality model. Autonomous Agents and Multiagent Systems **3**, 1217–1220 (2008)
9. Durupınar, F., Pelechano, N., Allbeck, J., Güdükbay, U., Badler, N.: How the ocean personality model affects the perception of crowds. IEEE Computer Graphics and Applications **31**(3) (2011)

10. Goldberg, L.: An alternative" description of personality": The big-five factor structure. Journal of personality and Social Psychology **59**(6), 1216 (1990)
11. Gröschel, A.: Towards believable crowd simulation for interactive real-time applications. Thesis, Hochshule fur Technik und Wirtshaft Berlin (2011)
12. Guy, S., Chhugani, J., Curtis, S., Dubey, P., Lin, M., Manocha, D.: Pledestrians: A least-effort approach to crowd simulation. ACM SIGGRAPH/Eurographics Symposium on Computer Animation pp. 119–128 (2010)
13. Harris, T.: I'm okay, you're okay:a practical guide to transactional analysis. Harper Perennial (2004)
14. Helbing, D., Molnar, P.: Social force model for pedestrian dynamics. Physical review **51**(5) (1995)
15. James, J.: The distribution of free-forming small group size. American Sociological Review (1953)
16. Jan, D., Traum, D.: Dialog simulation for background characters. Intelligent Virtual Agents pp. 65–74 (2005)
17. Jan, D., Traum, D.: Dynamic movement and positioning of embodied agents in multiparty conversations. Proceedings of the Workshop on Embodied Language Processing pp. 59–66 (2007)
18. Karamouzas, I., Overmars, M.: Simulating the local behaviour of small pedestrian groups. Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology pp. 183–190 (2010)
19. Moussaïd, M., Perozo, N., Garnier, S., Helbing, D., Theraulaz, G.: The walking behaviour of pedestrian social groups and its impact on crowd dynamics. PLoS One **5**(4), e10,047 (2010)
20. Musse, S., Thalmann, D.: A model of human crowd behavior: Group inter-relationship and collision detection analysis. Computer Animation and Simulation **97**, 39–51 (1997)
21. Ondřej, J., Pettré, J., Olivier, A., Donikian, S.: A synthetic-vision based steering approach for crowd simulation. ACM Transactions on Graphics (TOG) **29**(4) (2010)
22. Padilha, E., Carletta, J.: A simulation of small group discussion. Proceedings of EDILOG pp. 117–124 (2002)
23. Patel, J.: Simulation of small group discussions for middle level of detail crowds. DTIC Document (2004)
24. Pedica, C., Högni Vilhjálmsson, H., Lárusdóttir, M.: Avatars in conversation: the importance of simulating territorial behavior. Intelligent Virtual Agents pp. 336–342 (2010)
25. Pedica, C., Vilhjálmsson, H.: Spontaneous avatar behavior for human territoriality. Intelligent Virtual Agents pp. 344–357 (2009)
26. Pelechano, N., Stocker, C., Allbeck, J., Badler, N.: Being a part of the crowd: towards validating vr crowds using presence. Autonomous Agents and Multiagent Systems pp. 136–142 (2008)
27. Popelová, M., Bída, M., Brom, C., Gemrot, J., Tomek, J.: When a couple goes together: Walk along steering. Motion in Games pp. 278–289 (2011)
28. Reynolds, C.: Steering behaviors for autonomous characters. Game Developers Conference (1999)
29. Ricks, B., Egbert, P.: Improved obstacle relevancy, distance, and angle for crowds constrained to arbitrary manifolds in 3d space. Eurographics (2012)
30. Ricks, B., Egbert, P.: More realistic, flexible, and expressive social crowds using transactional analysis. Computer Graphics International **To Appear** (2012)
31. Scheflen, A., Ashcraft, N.: Human territories: How we behave in space-time. Prentice-Hall (1976)
32. Singh, S., Kapadia, M., Faloutsos, P., Reinman, G.: Steerbench: a benchmark suite for evaluating steering behaviors. Computer Animation and Virtual Worlds **20**(5-6), 533–548 (2009)
33. Yeh, H., Curtis, S., Patil, S., van den Berg, J., Manocha, D., Lin, M.: Composite agents. Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation pp. 39–47 (2008)