35 A Simple and Practical Social Dynamics System

Phil Carlisle

35.1 Introduction

One of the issues with current games that have large numbers of characters is that they often do not portray many of the coordinated interactions seen in any social group in the real world, for example, when friends cluster together at parties or when long-separated relatives hug each other after finally meeting each other again. Social interactions are an important part of how we understand the structure of social groups, and we would be wise to portray them in order to make the experience of the world more compelling.

This article will describe a social dynamics implementation based on modular components that together form a system that enables characters to take part in and portray social interactions. Building on a foundation of commonly used components such as behavior trees, blackboards, animation, and locomotion controllers, we will discuss aspects of nonverbal behavior commonly seen in social situations and will provide source code examples and practical implementation details (source code available on the book's website: http://www.gameaipro.com).

35.2 The Importance of Observation

When animators work on characters, they do so by implementing motions that they have observed, either through their own experience or from seeing other similar characters.

Similarly, many of the methods implemented here are intended to reproduce aspects of social interactions observed by both academics and animators. At the foundation of all of this work is the notion that observation of interactions is key to our being able to implement them. We must study life if we are to approximate the illusion of life in our characters.

AI programmers who are interested in working towards believable characters are strongly urged to become keen observers of human interaction. It is relatively easy to spot subtle interactions that can help sell any given relationship or social dynamic, especially if we film a number of such interactions and later analyze the footage. Small digital cameras that are suitable for capturing social interactions without being observed are relatively cheap and are a powerful tool to have available when working on any given scene.

35.3 What Is a Social Dynamic?

A social dynamic is any social interaction that has to happen in real-time between two or more characters. Typically, there is an element of spatial position, timing, and orientation involved in the interaction, hence the term "dynamic." There are a number of different elements that could be classified as social dynamics, and each of them can contribute to a more believable set of social interactions for characters, which will ultimately add to the believability of the world.

Typical examples are:

35.3.1 Gaze Control

We learn a lot about characters by looking at their face. One of the key things we understand is that if a character is gazing (looking) at an object, then it is likely aware of that object. The other aspect of gaze is that it focuses our attention on what is important to the character. For example, we can understand if one character is interested in another if their gaze is held for any significant time. This is useful to signal to the player that a character is attracted to another. Similarly, if a character we see suddenly gazes in a given direction, it is likely that we should pay attention in that direction, too, which can be useful for leading the players' view towards a particular visual event.

35.3.2 Proxemic Control

When we interact with other humans, we tend to keep a specific distance from them, depending on how well we know them, whether we like them, etc. This field of study is known in social psychology as *proxemics* and is important because it gives us a model of how humans move around within any social group.

As humans, we have a preferred distance which we maintain during different social interactions. For instance, when chatting socially, we have a relatively relaxed distance, but when trying to be intimate with someone we generally get a lot closer, often within easy touching distance. We should pay attention to proxemics because it gives us a general guide for forming small social groups, especially in spatial terms. Readers are referred to a useful paper on the application of this proxemic distance for use in games by Claudio Pedica [Pedica and Vilhjálmsson 08].

35.3.3 Posture

When interacting with another, we often adopt a given posture, depending on the nature of the interaction and the relationship we have with them. In a context where we are chatting with friends, we might adopt a relaxed posture where our arms are by our sides or gesturing. We might have a wider stance, too, and we will often lean towards people we are attracted to. Conversely, if we encounter someone in a position of authority over us, or in a role that puts social pressure on us to act with restraint, we might show this by having a more closed posture, with feet planted more firmly and closer together, our back straight or even leaning away slightly. These relatively subtle changes in posture can be used to signal differences between characters during social interactions. In addition, our posture can also affect elements of our gait when walking. Animators often exaggerate certain motions to affect changes in posture that imitate good or bad moods, for instance.

35.3.4 Gesture

This is perhaps the most challenging aspect of social dynamics in that it appears simple. It is easy to simply play gesture animations on a character, but the underlying reasons for why we gesture and what gestures we make are quite complex. The biggest area where we have problems in games is in the area of coordinated character gestures. If we observe real world social interactions we see many examples of gestures where one character touches another. A simple greeting might result in a handshake for instance. Yet in games, coordinating animations is actually quite difficult, not least because of the cost of animating a wide enough range of gestures to allow coordination to occur.

Readers are advised to review literature in the area of embodied conversational agents for more information on a number of posture and gesture studies; a good reference is the book *Embodied Conversational Agents* [Cassell 00].

We also note that the stated social dynamic elements are not an absolute requirement for every situation. They should be considered as extra details that provide subtle but useful hints to the player, much in the same way that additional texture data is used to add detail to rendered objects. However, it is likely that future games will feature more depth of social interaction as we develop our understanding of how this affects player perception.

35.4 Implementation

Rather than attempt to create a single system that portrays all social dynamics, instead we construct a number of systems that deal with individual aspects of social behavior and rely on the composition of these systems to implement the whole. In the example code, you will notice that all entities are simply composites. Please refer to the companion chapter number 34 “A Simple and Robust Knowledge Representation System” for more detail on the component-based architecture used.

The component implementations for the social dynamics system fall roughly in line with the aspects of social dynamics discussed previously. Components for gaze, proxemics, posture, and gesture control are simply added to characters at run time during instantiation of the character template. Where possible, a component performs a narrow subset of behaviors without requiring aspects of other components, but in the case of many of these social behaviors, other components are required. Frequently, social components delegate actions to other components. For instance the `ProxemicComponent`, which controls the social distances at which a character interacts, requires that there is a `LocomotionComponent` or other movement oriented component available in order to request the character to change position.

Before we describe the various components involved in the social dynamics system, we should describe how they are coordinated. Some aspects of social interactions are entirely based around the individual involved. For instance, an individual chooses the focus of their gaze and thus their attention. Yet most social interactions involve dynamically reacting to another person. These interactions can often involve groups of characters that may change over time. A chat may start out with two or three characters, expanding to five or six as more join the group, eventually having the original characters leave the chat, leaving none of the original group members involved. The initial group members instigated a social interaction that outlasted the participation of the instigators. This leads to the realization that another entity must be instantiated to monitor and control participation in the group activity.

35.4.1 SocialObjectComponent

This component performs the task of coordinating much of the group formation aspect of the system. At its core, it is a component that handles set membership, allowing characters to request access to the group, removing characters that are no longer participating, and allocating resources and/or positions in the group structure. This system is also responsible for advertising the availability of the social interaction as well as organizing flow control for when resources are limited, which as an example is useful to control how many people are talking at once during a group discussion.

This `SocialObjectComponent` is usually either added to the world during instantiation of an object or it is added dynamically during the update of a character that is receptive to a social encounter. An example of the former is a hot dog stand, where the `SocialObectComponent` is instantiated to control the behavior of the characters as they use the stand to buy hot dogs. An example of the latter would be when a character has true conditions for `<idle>`, `<wants_social>`, `<sees_friend>`, and `<friend_also_wants_social>`. When the conditions for social activity are met, the character spawns a `GameObject`, which has a `SocialObjectComponent` added. This becomes a proposal for social interaction and the `SocialObjectComponent` begins its role in coordinating the interaction.

35.4.2 SocialComponent

The intracharacter coordination of the various social dynamics components is controlled by the `SocialComponent`. This component is responsible for querying the world as to available potential interactions, forming requests to participate, controlling the focus of attention, etc. Much of the work of this component is involved in handling events propagated through the game and sending events to the different components of the social dynamics system to handle. For instance, the social component sends an event to its parent `GameObject` to indicate that the attention of the character has changed.

35.4.3 GazeComponent

This is perhaps the easiest component to implement in that it functions as a simple modifier to the animation component. Adding a `GazeComponent` to a character's xml template schema will mean that the component first initializes itself by requesting access to the `AnimationComponent` of the character `GameObject`. If this access fails, the `GazeComponent` asserts the failure, alerting the programmer to the dependency. The next step is for the `GazeComponent` to add itself as a listener of the `AnimationComponent`, which allows it to alter the animation prior to the animation being submitted for rendering the final character. During the listener callback method, the `GazeController` simply changes the orientation of the head bone within specific limits, along with the spine bones as in Figure 35.1.

Please refer to the method `UpdateGaze()` for the actual math involved in modifying the animation, but it is perhaps useful to note that for a character with more spine nodes, the rotation of each spine bone should be scaled the further away from the head bone they are in the hierarchy. This allows for some amount of torso twist, which is desirable to mimic the twist available in the human torso. Obviously the number of spine links in a typical game character is less than the number in a human so there will always be some visual discontinuity, but in general some torso twist is enough to sell the motion. It should be noted that the amount of rotation that both torso and head bones are allowed should be carefully selected so that they are in a range of motions that would be generally considered comfortable for any given character type, depending on age, build, etc.

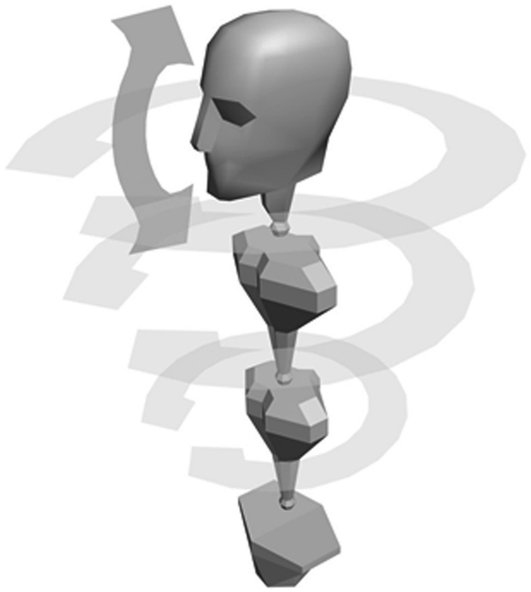

Figure 35.1

The skeleton of a character is manipulated by rotating the bones of the head and spine in order to allow the head to "look at" a given position. Note the restricted range of the vertical and horizontal motion (horizontal restriction not shown) and the reduced proportion of the horizontal motion as we get further away from the head.

The `GazeComponent` reacts to events sent via the `SocialComponent` to "look at" another entity and/or position. The duration of the gaze is modified by the intensity of the interaction as noted in the "look at" event. Eventually the gaze controller resets the gaze direction, or may even modulate the current gaze direction to temporarily look away from the gaze target for a brief period. The character personality data specified in the `AIComponent's` blackboard allow for the gaze to be modified such that a shy character looks away more often than an assertive one.

35.4.4 The Social Origin

A key aspect of coordinating movement for social interactions is having a shared social origin. It is useful to think of many social interactions as a spatially oriented set of timed actions; in order to have coherent animations, we must assume that the social interaction is performed with respect to a shared origin. This allows for each interaction to function anywhere in world space as coordination acts in a space relative to the social interaction which is propagated to all participants.

This role of coordinating the shared origin is part of the `SocialObjectComponent` class as part of its responsibility as arbiter of the interaction. Individual characters involved in the interaction must respect this shared origin as they calculate their own movements while also respecting the movements of other characters. This approach bears some resemblance to the moving origin techniques used for animating a character during parkour-like behavior. For more information see an interview with Laurent Ancessi of Naughty Dog on AIGameDev.com [Ancessi 10].

35.4.5 ProxemicComponent

Controlling the proximity to other characters during social interaction requires access to the `LocomotionComponent` of the character (or other functionality which serves to move the character and orient them in space). The `ProxemicComponent` request is constrained by the locomotion, navigation, and collision avoidance strategy of the character movement. The `ProxemicComponent` is responsible for calculating the ideal position of the character during any movement of agents in the social group.

This means that, for instance, if a character leaves the group, the other characters may change position in order to stay in reasonable proximity for the social interactions. In practice, this means that each character tries to maintain a comfortable proxemic distance from all other characters in the group, while still maintaining other constraints such as line of sight or being close enough to touch with gestures for characters that have positive affection. Acting in a very similar manner to steering behaviors, the proxemic distance is maintained via a simple vector length calculation which relates character distance away from a group circle. This behavior approximates what has been observed by social scientist Adam Kendon who referred to the phenomenon as an "O frame" [Kendon 90], although it must be noted that it is useful to dampen any movement force such that the character is not continually shifting position (Figure 35.2).

35.4.6 PostureComponent

Posture is one of the easier components to implement as it simply relies on animation selection and/or blending to achieve the desired effect. The implementation provided demonstrates the simplest form of this animation selection to simply affect a bias in the

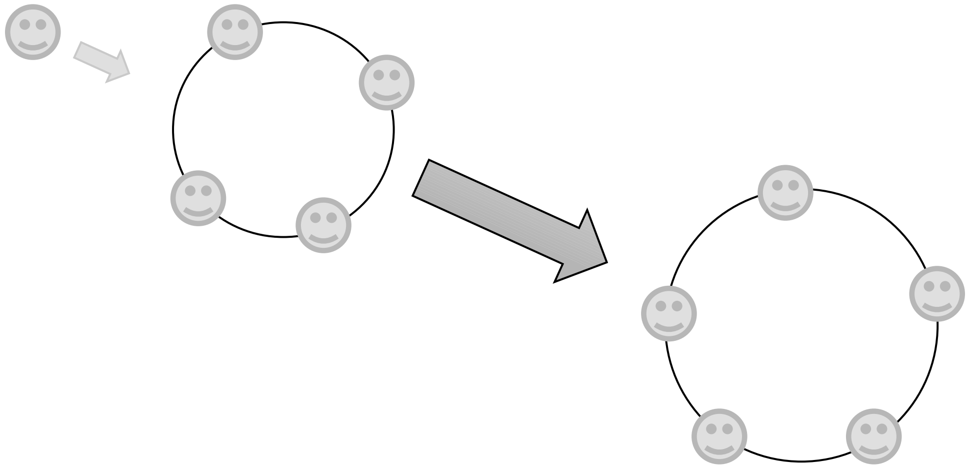

Figure 35.2

Proxemic distance is controlled by the `ProxemicComponent` that calculates a position offset relative to a circle known as an "O frame" and attempts to maintain that position as members of the group change.

choice of available animations. A more complex implementation would be to add blend nodes into an animation blend tree that would blend different postural clips at different weights based on the overall mood of the character. This requires that a number of similar animation clips be prepared that correspond to the changes in mood that would normally affect posture.

Effects such as fatigue, happiness, sadness, and excitement can then be portrayed by simply selecting the appropriate clip from the available animation set. In the case of the Floyd character in the demonstration provided, there are a number of animation clips that are chosen based on a notional "mood" value held in the blackboard of the character. With this we are able to portray a number of moods through postural changes that can be quite subtle; elements such as slumping of the shoulders, making the eyelids droop, or in the case of Floyd, who is intended as a comedic sidekick worker robot, his "eye" glows slightly less to denote a more somber mood.

35.4.7 GestureComponent

Gestures play a large part in the portrayal of a character, in that they allow the player to quickly appraise the feelings of the character by observing the gestures made. Characters that gesture often, with wide arm movements and large shifts of body weight, are generally considered more energetic and positive. Once again, the `GestureComponent` essentially modifies the animation of the character by selecting different animation clips. In this case, the clips are usually additively blended on top of basic motion clips. In the case of a humanoid character we must take care not to attempt a gesture while performing other animations that would look out of place. So, for instance, we only play gestures that require arm movements when no other animation is playing that would affect the arms too strongly. More specifically, it makes no sense to incorporate greeting gestures when the character is doing a forward roll.

One of the more challenging aspects of the `GestureComponent` functionality is in knowing when to initiate a gesture. Typically, we gesture more when we are trying to make a point or guide a conversation. Usually this gestural "language" underlies a discussion by punctuating key words with gestures, known by social scientists as *nonverbal communication*. Fully describing the role of nonverbal communication is beyond the scope of this chapter, but it is very important for believable characters. The reader is recommended to seek out academic work in this area such as *Bodily Communication* by Michael Argyle [Argyle 88].

Given that gestures often accompany speech, it may be useful to allow audio engineers to trigger gestures by allowing them to send events at appropriate points in the audio clip. Another issue is when gestures are required to touch another character. This requires a great deal of precision in order to achieve the touch without problems of penetrating the mesh of the other character. Although it is not implemented in the example code, a simple 2 bone inverse kinematic controller is often used to control the exact position of the hand during gestures involving other characters.

While the class diagram in Figure 35.3 may seem complex, the beauty of this system of components is that each component is relatively simple to implement. Because each component deals with a single aspect of behavior, we can structure the code to be straightforward in dealing with only that single aspect. In practice, this means that we can implement each component optionally; components are tied together via events and generally do not know about one another or rely on each other to function. The exceptions to this case are that

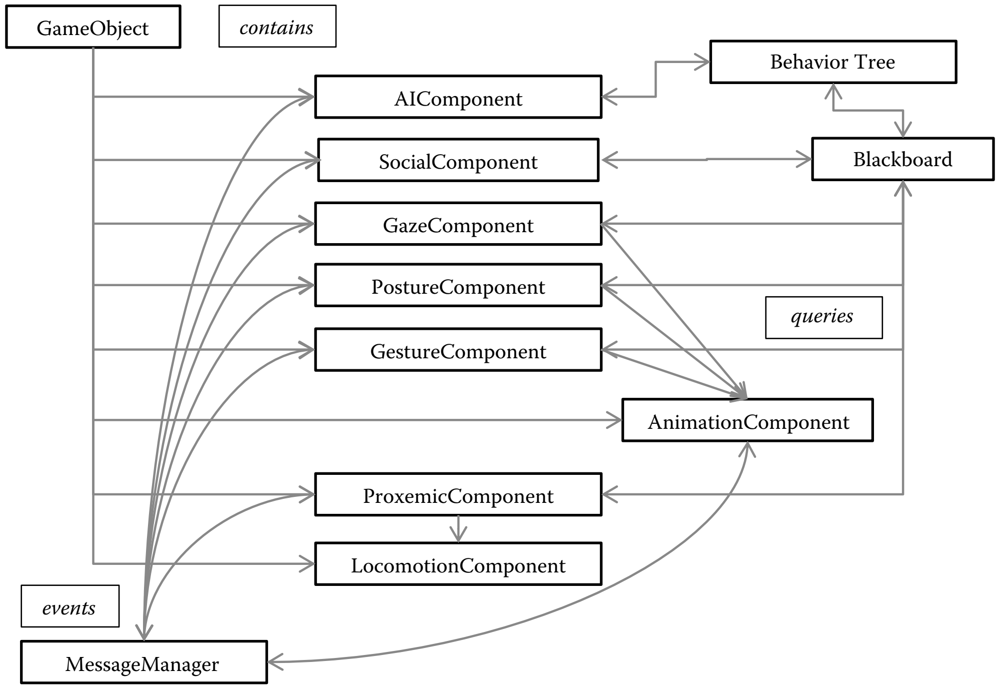

Figure 35.3

Shows how the individual components communicate with one another and with the behavior tree and blackboard implemented within the AIComponent.

most of the components discussed require access to the AnimationComponent and that the functioning of many of the components requires that the SocialComponent be added to the character in order to coordinate the behavior. Even this is not a strict requirement, as it is possible to debug behavior by simply injecting the relevant events into the event stream to be read by the various components.

35.5 Execution

As is shown in Figure 35.3, the individual components are contained within a GameObject parent container. Each component can access data from the AIComponent via the blackboard. When a character becomes aware of available social interaction possibilities, either via the sensory system or via the event system, this information is placed into the blackboard. This in turn makes the behavior tree conditions become true, which cause an event to be sent to the SocialComponent of the parent GameObject. The SocialComponent in turn sends an event to the SocialObjectComponent of the sensed object to request participation in the interaction.

Once the SocialObjectComponent receives enough requests to fulfill its role, it sends an event to all participants notifying them of their role in the interaction. Prior to this notification, all agents are continuing with their previous behavior. It should be noted that the number of potential social interactions in which a given agent can request participation should be determined by the requirements of the game and can have an impact on overall performance. Too few requests mean agents appear to be unsociable, but too many requests create unnecessary processing to occur when agents have to remove their requests upon successfully starting an interaction. Events to set the origin for the interaction, as well as participant information, turn taking, and other coordination information, are sent periodically. Finally when the interaction is deemed completed, the SocialObjectComponent sends an event to release all remaining participants, allowing them to continue seeking other interactions or to pursue alternative behavior.

Each update of the AIComponent in the main game loop executes the character behavior tree, which in turn updates data in the blackboard. During the same update loop, the SocialComponent reads the updated data and sends events to the other components that respond with appropriate changes in position/orientation/posture/gaze/etc. The final effect of these changes is then either assimilated into the current animation or is output as forces which are then incorporated into the next locomotion update.

35.6 Conclusion

The requirement for social dynamic behavior for game characters is compelling. As we strive for ever increasing visual realism, we should also strive for behavioral realism. This is not to say that we need to restrict ourselves to "realistic" behavior so much as to propose that we pay attention to the facets of behavior that increase the believability of our characters. A component-based approach to these facets of behavior allows us to iteratively implement and refine our approaches to these aspects. We can start off with a simple "look at player" component and extend the system over time. The goal is to create characters that enable the player to believe they are alive and ensuring that they play their part in making the game a compelling experience. Incorporating the finer details of character social dynamics helps lead us to worlds in which the player believes in the illusion of life.

References

[Ancessi 10] Laurent Ancessi interview with Alex Champandard (aiGameDev.com). Available online (http://aigamedev.com/premium/masterclass/interactive-parkour-animation/).

[Argyle 88] M. Argyle. *Bodily Communication*. Taylor & Francis, 1988, p. 363.

[Cassell 00] J. Cassell. *Embodied Conversational Agents*. MIT Press, 2000, p. 440.

[Kendon 90] A. Kendon. *Conducting Interaction: Patterns of Behavior in Focused Encounters (Studies in Interactional Sociolinguistics)*. Cambridge University Press, 1990, p. 308.

[Pedica and Vilhjálmsson 08] C. Pedica and H. Vilhjálmsson. "Social perception and steering for online avatars." *Intelligent Virtual Agents*, Springer, 2008, pp. 104–116.