Understanding and Improving Collaborative Skills Among Individuals with ASD in a Distributed Virtual Environment

Arpan Sarkar[1], Joshua Wade[2(✉)], and Zachary Warren[3,4]

[1] University School of Nashville, Nashville, USA
[2] Electrical Engineering and Computer Science, Nashville, USA
joshua.w.wade@vanderbilt.edu
[3] Treatment and Research Institute for Autism Spectrum Disorders (TRIAD), Nashville, USA
[4] Pediatrics, Psychiatry and Special Education, Vanderbilt University, Nashville, TN 37212, USA

Abstract. Individuals with Autism Spectrum Disorders (ASD) evidence core impairments regarding social interaction and communication. These impairments can inhibit the ability of individuals with ASD from effectively engaging with peers and collaborating on goal-oriented tasks. Recently collaborative virtual environment (CVE) in which individuals with ASD can interact with one another or with a therapist to achieve some common goal has been proposed for social competence interventions (SCI) for these individuals. In this paper, we present the design of a distributed CVE for playing the classic video game pong to be used for SCI. This collaborative game can be played at several different modes ranging from one player against an artificial agent in one computer to two players against each other in two different computers. The system functionality and robustness were validated through a small user study. In the future, this CVE will be evaluated with children and adolescents with ASD.

Keywords: Collaborative Virtual Environment (CVE) · Autism Spectrum Disorder (ASD)

1 Introduction

The Centers for Disease Control and Prevention estimates 1 in 68 children in the United States has an Autism Spectrum Disorders (ASD) [1]. Individuals with ASD evidence core impairments regarding social interaction and communication [2–6]. These impairments can inhibit the ability of individuals with ASD from effectively engaging with peers and collaborating on goal-oriented tasks, which in terms can contribute to further social isolation, distress, and impairment. Various implementations of social competence interventions (SCI) have emerged that try to address this problem. Recently, many of these interventions have taken the form of a collaborative virtual environment (CVE) in which individuals with ASD interact with one another or with a therapist to achieve some common goal [7–11]. SCIs based on CVE technology are highly flexible and have some advantages over alternative intervention methods.

DOI: 10.1007/978-3-319-20684-4_64

Users of CVEs may communicate verbally or non-verbally with one another [12] with no restrictions on the physical distance between individuals and administrators as there are in clinic-based interventions. Personal interaction in CVEs may feel “safer” than in the real world for individuals with ASD [11], and CVEs are capable of creating interactive scenarios that would not otherwise be possible in the real world.

Recent work on SCIs using CVEs have utilized the strengths of CVEs to deliver targeted interventions to individuals with ASD. Stichter et al. [7] deployed iSocial: a distributed, 3D virtual learning environment which was designed as a SCI for individuals with ASD in a distance learning format. ISocial contained collaborative tasks including one in which students worked together to design and build a restaurant in the virtual world, requiring students to talk with one another and coordinate their actions in order to complete tasks. Hourcade et al. [8] designed an array of tablet-based activities designed to encourage social interaction and face-to-face collaboration among children with ASD. The activities provided on the tablets included drawing (to create and/or express how they feel), music authoring (to create different kinds of media), puzzle-solving (to test communication, collaboration, coordination, and visuo-spatial reasoning), and a photo-distorting app (to explore emotions on faces). Cheng et al. [10] developed a CVE aimed at increasing empathy in individuals with ASD by showing them scenarios intended to elicit empathic responses such as witnessing a person cutting in line or slipping and falling on the floor.

In this work, we present the design of a distributed CVE based on the game pong to understand and eventually improve the collaborative skills of individuals with ASD. The system that we designed records quantitative, objective data from users regarding measures of performance—both individual and collaborative—as well as conversation through recorded audio.

In this paper, we discuss the design decisions made in the development of this system as well as a series of validation tests of the system to ensure its correctness and efficiency. The remainder of the paper is organized as follows. Section 2 details the hierarchical, hybrid automata model of computation used in implementing the software. Section 3 discusses the preliminary evaluation of the system with volunteers. In Sect. 4, we provide the results from both the throughput analysis and the validation study. Section 5 concludes the paper with a discussion of the limitations of the current system and proposed future work.

2 System Design

2.1 Overview

Pong is a classic video game with a simple interface and is based on the game of ping pong in which two players use paddles to hit a ball back and forth trying to cause their opponent to miss the ball. Figure 1 shows an example pong interface. We selected pong as the basis for a collaborative game because it is simple to understand, easy to implement, and offers a variety of ways in which collaboration can be introduced into the game. We chose to implement the element of collaboration through the control of a single game paddle by more than one user—two users in this case. The subsections that follow elaborate on the design of gameplay, overall architecture, models of computation used to create system components, and our method of converting the models into software.

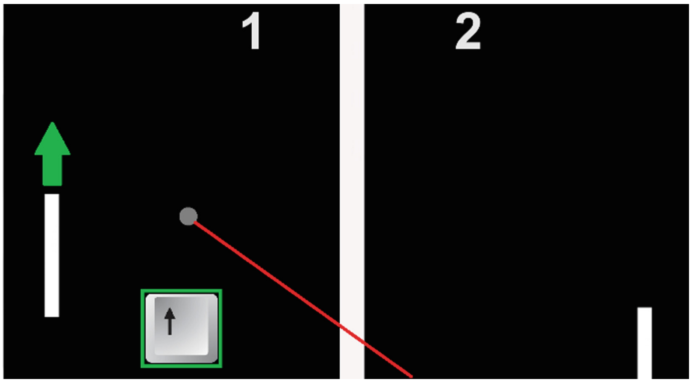

Fig. 1. Basic pong interface

2.2 Game Design

Our version of the pong game consists of a collection of *matches* which we refer to as a *session*. A match is defined as an interval of time beginning when the ball is deployed and ending when one player misses the ball. Sessions can be played in one of several different types of *configurations*. The following configurations were implemented: (1) single human *vs*. artificially intelligent agent (AI) on a single PC, (2) human team *vs*. AI on a single PC, (3) human team *vs*. AI on separate PCs, (4) human *vs*. human on a single PC, and finally (5) human *vs*. human on separate PCs. Matches in a session are presented sequentially with a 3 s countdown before the match starts and a period of feedback following the end of a match. This feedback is presented as both text and audio and either congratulates the player for scoring, encourages them if they do not score, or remains neutral in the event of a draw.

The AI was designed with three different difficulty settings: easy, medium and hard. In the easy setting, the AI follows an oscillating trajectory based on the triangle wave function with a period of 2 s (1).

$$p(t) = 2(-1)^{\left|t+\frac{1}{2}\right|}\left(t - \left|t + \frac{1}{2}\right|\right) \tag{1}$$

This behavior is considered easy because the AI effectively demonstrates random movement without any consideration of the ball's trajectory. In the medium and hard settings, the AI predicts the destination of the incoming ball and moves towards this goal with a speed controlled by a scalar k and a linear interpolation function l_2, where the value of k is larger in the hard setting (i.e., faster) and smaller in medium setting (i.e., slower). In order to guarantee that the AI fails to reach the goal occasionally, a random error within a reasonable range was applied to the calculation of the goal position.

2.3 Architecture

There are several components of this work which comprise the overall application architecture. Figure 2 gives a representation of the architecture as a block diagram where each block is a separate component. When the application runs, a server or client role is specified. The blocks shaded in grey are executed only when the role chosen is that of the server and are disabled otherwise. The Main Controller is the central component of the application—regardless of role—and handles synchronization of all the adjacent components. The Network Module takes on the behavior of either a server or client and manages all sending and receiving of messages over the network. The messages passed over the network are objects serialized as JSON strings. Messages sent from the client to the server contain input from the human player while messages from the server to the client contain the current state of the CVE. The Session Manager executes in either server or client role and ensures that the client instances of the CVE reflect the true state of the server instance. In the System Configuration Module, information regarding IP addresses and port numbers as well as the session configuration to execute is specified.

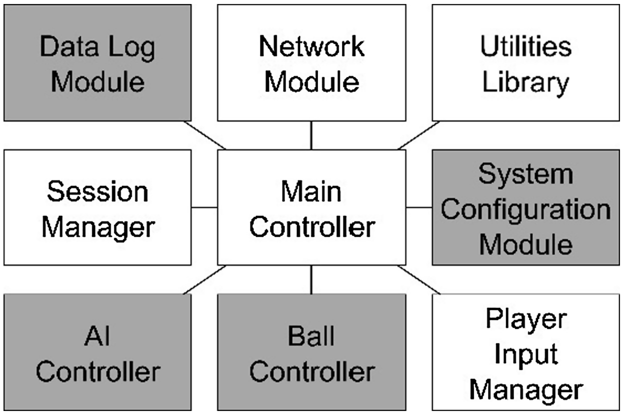

Fig. 2. Block diagram of the overall architecture (server-exclusive components shaded in grey)

The remaining components' functions should be evident based on their names. The Data Log Module records several different types of data. Time series data including the velocity and position of the ball, velocity and position of both paddles, and player input are recorded with a timestamp at the fastest possible rate. Data related to various predefined events are also logged such as at the beginning and ending of matches, at the beginning and ending of feedback, and for collisions of the ball with other objects. In addition, audio is recorded from two separate microphones for offline analysis of speech. The Ball Controller simply updates the position of the ball during matches while the AI Controller updates the position of the AI player. The Player Input manager obtains player input on the client side and applies changes to the player's paddle on the server side.

2.4 Models of Computation (MoC)

The nontrivial components of the system were first modeled using formal models of computation (MoC) and then converted to software. The latter step is discussed in the next subsection. The MoCs chosen were finite state machines (FSM) and hybrid automata (HA). More specifically, the FSMs in this system are composed hierarchically in what is referred to as a hierarchical state machine (HSM). The syntaxes for FSMs, HSMs and HA used here are in the style found in Lee and Seshia [13]. Since there are so many components of this architecture, a full description of each one is not possible, so we instead focus on the MoCs involved in the AI player's behavior.

At a certain level of abstraction, the AI has only three distinct states: (1) idle because the match has not started, (2) idle because the match configuration type is human *vs*. human, and (3) playing. This behavior is captured formally in the FSM shown in Fig. 3. The initial state *GameNotStarted* indicates that a match has not yet begun and the AI is idle. The state *GameStartedAgentPlaying* is arrived at when a match begins *and* the match configuration type is not human *vs*. human. Conversely, the FSM transitions to *GameStartedAgentNotPlaying* if a match begins and the match configuration type *is* human *vs*. human. If a match ends while the FSM state is either *GameStartedAgentPlaying* or *GameStartedAgentNotPlaying*, then the FSM transitions back to the *GameNotStarted* state and remains there until the next match begins.

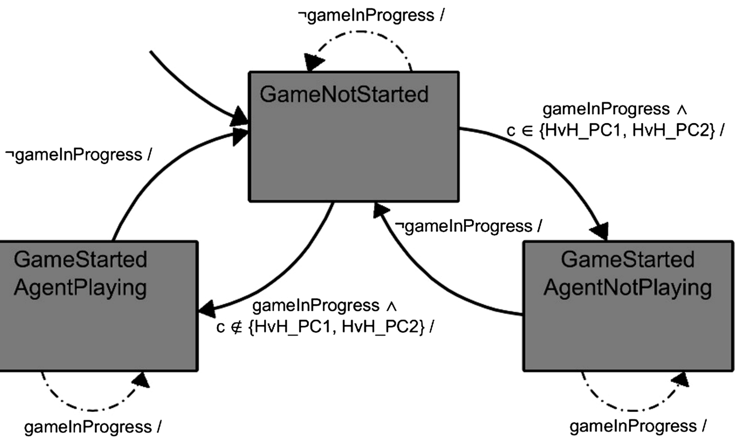

Fig. 3. FSM defining the general behavior of the AI player

Of course, this level of abstraction does not capture all of the complex behavior of the AI such as how it behaves under different difficulty settings or when the ball's trajectory changes and the AI must re-calculate its goal position. For this level of detail, we designed the HA shown in Fig. 4 which is an expansion of the AI's FSM state *GameStartedAgentPlaying*. Upon entering the HA in Fig. 4, a junction determines which transition is taken first based on the specified difficulty level *d*. If *d* is equal to the value *EASY*, then the transition to the discrete mode named *Oscillating* is taken and the AI follows the triangle wave trajectory described earlier. While in this mode, the velocity of the AI is constant [$v(t) = 1$] and—as with all of the discrete modes in this particular HA—the rate of change of time is assumed to be constant as well [$\dot{s}(t) = 1$].

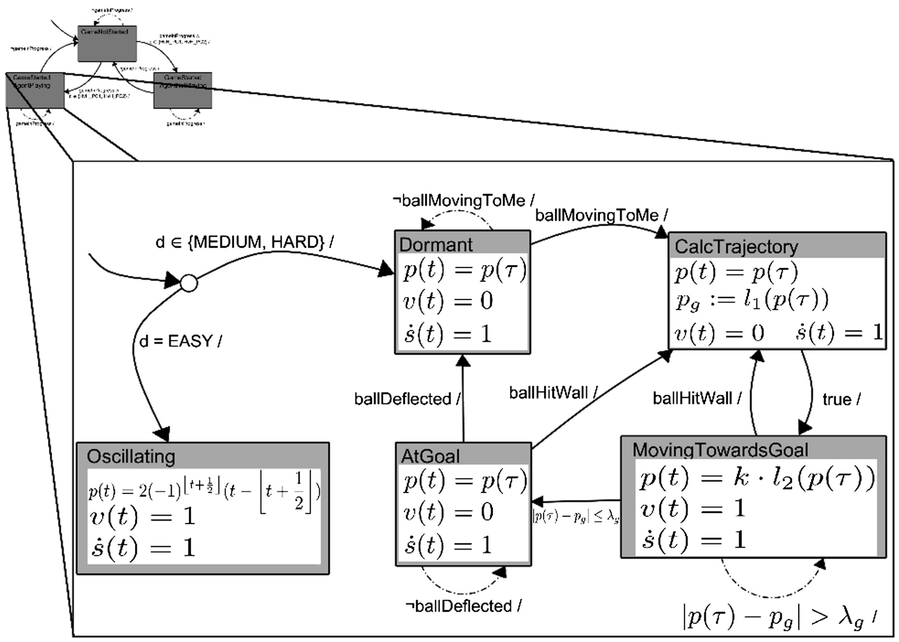

Fig. 4. Hybrid Automaton (HA) describing movement of AI player. This HA is an expansion of the more general FSM shown in Fig. 3.

If the value of *d* had instead been *MEDIUM* or *HARD*, then the transition to the discrete mode *Dormant* would have been taken where the AI would remain idle [$v(t) = 0$] until the ball began to move towards it. When it was detected that the ball was moving towards the AI player, the transition from *Dormant* to *CalcTrajectory* was taken. Within *CalcTrajectory*, the AI would predict—with some random error—the path of the ball based on the ball's position and velocity via the function l_1, and would then transition to the mode *MovingTowardsGoal*. While in *MovingTowardsGoal*, the AI moved with constant velocity towards the calculated goal position until it was within some small distance threshold λ_g of the goal. Once arrived at the goal, the AI would remain there until either the ball was deflected successfully or the ball's direction changed due to hitting a wall.

The hierarchy shown in Fig. 4 is only a small view of the total hierarchy involved in the system. At the highest level there is the Main Controller's FSM which includes states for obtaining the network role from the user as well as starting and stopping sessions. Below the Main Controller's FSM is the Session Manager's FSM whose states deal with setting up matches, performing the match countdown, executing the match, and giving feedback. All of the AI behavior is below the Session Manager's FSM. Next we describe how we implemented this 4-layer HSM in software.

2.5 Implementation

The authors used their own method of converting formal MoCs into programming code. A base class called *Automaton* maintained state variables and provided a state transition function:

```
public class Automaton {
  private int m_currState;
      ⋮
  public void Transition( int NextState ) {
    m_currState = NextState;
  }
}
```

Each formal MoC was then implemented as a derived instance of this *Automaton* object. For example, the HA shown in Fig. 4 was defined thusly:

```
public class MotionPlanningAutomaton : Automaton {
  public const int DORMANT = 0;
  public const int CALCULATING_TRAJECTORY = 1;
  public const int MOVING_TOWARDS_GOAL = 2;
  public const int AT_GOAL = 3;
  public const int OSCILLATING = 4;

  public MotionPlanningAutomaton() : base() {
    this.Transition(DORMANT); //default transition
  }
}
```

In most cases, each MoC was implemented in a separate file in order to maximize isolation which simplified the development process. The logic of each MoC was written in an if-then-else branching structure and was updated at the fastest rate that the system would allow. All of the code was written in C# using the Unity3D game engine (www.unity3d.com).

Figure 5 shows a stack of graphs which give the state values of the HSM from the AI's motion planning HA up to the Main Controller's FSM during a sample run of the system. A selection of discrete events is represented by the vertical lines. Table 1 gives details about the events as well as the responses of each relevant component of the system due to the events. The reader can see clearly from the graph that between events e_2 and e_3, the AI player is performing the triangle wave oscillating behavior. From event e_7 until the end of the graph, the AI player is using the intelligent method of play where it moves based on the ball's trajectory.

3 Procedures

3.1 Network Throughput Analysis

In order to assess the reliability and performance of the system, we analyzed the network throughput of the system defined here as the number of serviced requests from

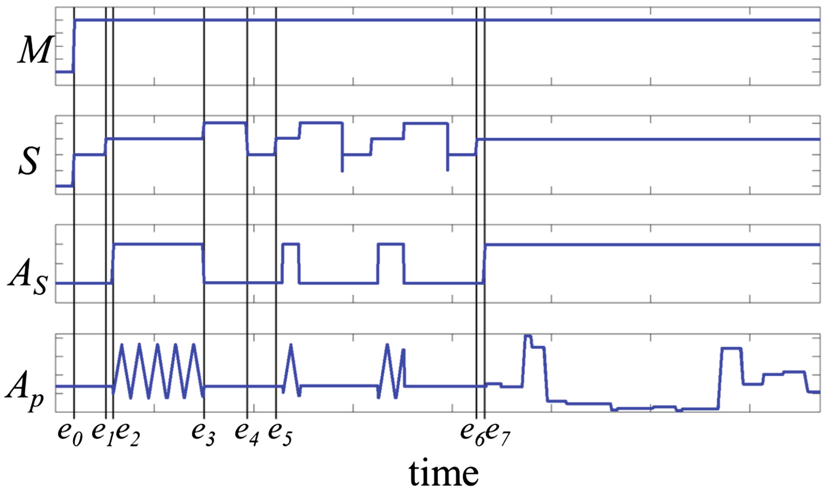

Fig. 5. AI Behavior HSM state values over time: the Main Controller FSM (M), Session Manager FSM (S), general agent FSM (A_S), and agent position (A_p).

Table 1. Events corresponding to HSM output graph in Fig. 5

Event	Description of system changes
e_0	***M***: session begins
	S: match countdown begins
e_1	***S***: countdown complete and match begins (*EASY* match)
e_2	$\boldsymbol{A_S}$: AI becomes active
	$\boldsymbol{A_p}$: AI begins **oscillating** trajectory
e_3	***S***: match ends and feedback begins
	$\boldsymbol{A_S}$: AI becomes inactive
	$\boldsymbol{A_p}$: AI stops moving
e_4	***S***: feedback done and countdown starts again
e_5	***S***: countdown complete and match begins (*EASY* match)
e_6	***S***: countdown complete and match begins (*MEDIUM* match)
e_7	$\boldsymbol{A_S}$: AI becomes active $\boldsymbol{A_p}$: AI begins **intelligent** trajectory

the perspective of the client(s) during regular gameplay. In order to provide a smooth experience to users, the update rate on the client side needed to be sufficiently fast so as not to show lag—in gaming, this value is often 60 frames per second (FPS), but can be as low as 30 FPS. We performed this analysis on two different network configurations: (1) a server and client running on the same PC, and (2) a server and client running on one PC while another client ran on a second PC. Note that there is no requirement that the server must be run on the same PC as any client. Both PCs had 3.7 GHz processors with at least 20 GB of RAM and NVIDIA Quadro K600 GPUs. Each configuration type was evaluated 5 times each and the results are presented in Sect. 4.

3.2 Human Validation Study

A small validation study was conducted with 2 pairs of volunteers—3 females (age, M 25.7 y, SD 1.7 y) and 1 male (age, 25)—to verify that the system was robust and complete. Each of the participants first completed a practice session consisting of 9 matches in which he/she played against the artificial agent to become acclimated to the system. Predetermined pairs of participants then completed a collaborative session on the same PC followed by a collaborative session on separate PCs. The goal was, contrary to the original pong game where each player plays to defeat his/her opponent, to play collaboratively with one's partner to defeat the AI player. All three sessions consisted of 9 matches with 3 easy, 3 medium, and 3 hard match difficulties. For side-by-side collaborative play, both participants sat at the same computer and shared a keyboard to control the paddle. One person used the up and down arrow keys of the keyboard to control the paddle while the other used the "W" and "S" keys. For collaborative play on 2 separate PCs, participants sat at adjacent desks and could speak to one another. In both types of collaborative play, 2 separate directional microphone devices captured the speech of the participants.

4 Results and Discussion

4.1 Network Throughput Analysis Results

Table 2 gives the results of the two analyses of the network throughput. In both cases, all of the involved clients demonstrated a throughput rate sufficiently high enough to avoid any noticeable image stuttering or lag. Since Client 1 ran on the same PC as the server in both evaluations, it had a very high throughput rate greater than 300 Hz while Client 2, which ran on a separate PC, had a throughput rate just under 60 Hz.

4.2 Human Validation Results

Table 3 gives the results of the user validation study for individual practice and collaboration for the two pairs of participants. Participants P0 and P1 performed comparatively well both individually and collaboratively. Nothing can be generalized from this small dataset, but it seems reasonable that player performance would drop from individual play to collaborative play since the sense of control is lessened. The player score indicates the number of times that the AI player failed to deflect the ball. Similarly, AI score indicates the number of times that the human player or players were unable to deflect the ball. A draw occurred if, after 1 min of play, no player, neither the human player(s) nor the AI player, scored.

Table 2. Network throughput rates in Hz

Trial number	Single client	Two clients	
	Client 1*	Client 1*	Client 2
1	340.14	319.85	59.83
2	327.25	309.86	59.65
3	339.94	318.63	59.84
4	344.12	275.08	59.88
5	341.55	332.48	57.20
Mean	338.60	311.19	59.28
SD	6.56	21.74	1.17

*Executed on the same PC as the server.

Table 3. User performance results

	P0	P1	P2	P3	P0, P1 Collaboration		P2, P3 Collaboration	
	Individual practice				Same PC	Diff. PC	Same PC	Diff. PC
Player Score	6	6	3	2	3	4	3	4
AI Score	3	2	5	7	3	3	6	4
Draws	0	1	1	0	3	2	0	1

4.3 Discussion

The network throughput analysis results provide significant evidence that the system is fast enough to handle one or two clients and therefore all of our proposed gameplay configurations. In fact, scaling up the number of clients in the future would likely contribute to relatively minor declines in system performance. The sample used for the validation test is obviously too small to draw any performance-related conclusions from, but it does show that our system is capable of collecting the data needed in order to assess whether an individual's performance varies when playing alone or in a team as well as whether the various configuration types affect performance.

5 Conclusion

We have developed a distributed CVE, based on the video game pong, to understand, analyze, and eventually address the collaborative and communicative skills of individuals with ASD. This system records quantitative and objective data regarding both individual and collaborative performance through analysis of recorded audio and gameplay.

There were some limitations regarding the experiment. First, we conducted a small user study with typically developed individuals to validate the system, but they are not the target group for this research. Second, we only implemented one form of input for the paddle (i.e., keyboard-based). And third, we used a wired connection for recording audio. In future work, we intend to address all of these issues. We plan to conduct a pilot user study on individuals with ASD, in order to determine differences in collaborative gameplay from that of typically developed individuals and also to determine any long-term effect that our system has on the collaborative skills of individuals with ASD. We intend to add other forms of input besides keyboard, such as joysticks or mobile devices (i.e., iPhone, Android, etc.). We also aim to implement some form of audio streaming to allow increased separation, while playing collaboratively, between participants in experiments. In addition, we intend to implement a "rally" mode which would allow a more sophisticated kind of collaboration game, instead of pure competitive gameplay. A rally mode would involve two players, each controlling their own paddle, playing with each other, instead of against each other, and the goal would be to continuously pass the ball from paddle to paddle. In this scenario, to succeed, users would have to accurately gauge their paddle movements and the effect their actions will have on the other player. We believe that this will provide highly accurate data regarding participation and collaboration between users.

Acknowledgment. This work was supported in part by the National Institute of Health Grant 1R01MH091102-01A1, National Science Foundation Grant 0967170 and the Hobbs Society Grant from the Vanderbilt Kennedy Center.

References

1. Wingate, M., Kirby, R.S., Pettygrove, S., Cunniff, C., Schulz, E., Ghosh, T., Robinson, C., Lee, L.C., Landa, R., Constantino, J., Fitzgerald, R., Zahorodny, W., Daniels, J., Nicholas, J., Charles, J., McMahon, W., Bilder, D., Durkin, M., Baio, J., Christensen, D., Van, N. Braun, K., Clayton, H., Goodman, A., Doernberg, N., Yeargin-Allsopp, M., Monitoring, A. D.D.: Prevalence of autism spectrum disorder among children aged 8 years - autism and developmental disabilities monitoring network, 11 Sites, United States, 2010. Mmwr Surveillance Summaries, vol. 63, March 28 2014
2. Senju, A., Johnson, M.H.: Atypical eye contact in autism: models, mechanisms and development. Neurosci. & Biobehav. Rev. **33**, 1204–1214 (2009)
3. Bekele, E., Crittendon, J., Zheng, Z., Swanson, A., Weitlauf, A., Warren, Z., Sarkar, N.: Assessing the utility of a virtual environment for enhancing facial affect recognition in adolescents with autism. J. Autism Dev. Disord. **44**, 1641–1650 (2014)
4. Bekele, E., Zheng, Z., Swanson, A., Crittendon, J., Warren, Z., Sarkar, N.: Understanding how adolescents with autism respond to facial expressions in virtual reality environments. IEEE Trans. Vis. Comput. Graph. **19**, 711–720 (2013)
5. Lahiri, U., Bekele, E., Dohrmann, E., Warren, Z., Sarkar, N.: Design of a virtual reality based adaptive response technology for children with autism. IEEE Trans. Neural Syst. Rehabil. Eng. **21**, 55–64 (2013)
6. Kuriakose, S., Sarkar, N., Lahiri, U.: A step towards an intelligent human computer interaction: physiology-based affect-recognizer. In: 2012 4th International Conference on Intelligent Human Computer Interaction (IHCI), pp. 1–6 (2012)
7. Stichter, J.P., Laffey, J., Galyen, K., Herzog, M.: iSocial: Delivering the social competence intervention for adolescents (SCI-A) in a 3D virtual learning environment for youth with high functioning autism. J. Autism Dev. Disord. **44**, 417–430 (2014)
8. Hourcade, J.P., Bullock-Rest, N.E., Hansen, T.E.: Multitouch tablet applications and activities to enhance the social skills of children with autism spectrum disorders. Pers. Ubiquit. Comput. **16**, 157–168 (2012)

9. Montoya, M.M., Massey, A.P., Lockwood, N.S.: 3D collaborative virtual environments: exploring the link between collaborative behaviors and team performance. Decis. Sci. **42**, 451–476 (2011)
10. Cheng, Y., Chiang, H.-C., Ye, J., Cheng, L.-H.: Enhancing empathy instruction using a collaborative virtual learning environment for children with autistic spectrum conditions. Comput. & Educ. **55**, 1449–1458 (2010)
11. Cheng, Y., Moore, D., McGrath, P., Fan, Y.: Collaborative virtual environment technology for people with autism. In: Fifth IEEE International Conference on Advanced Learning Technologies, ICALT 2005, pp. 247–248 (2005)
12. Nguyen, T.T.H., Fleury, C., Duval, T.: Collaborative exploration in a multi-scale shared virtual environment. In: 2012 IEEE Symposium on 3D User Interfaces (3DUI), pp. 181–182 (2012)
13. Lee, E.A., Seshia, S.A.: Introduction to embedded systems: A cyber-physical systems approach. Lee & Seshia (2011)