Towards Emergent Play in Mixed Reality

Patrick Misteli
ETH Zurich
pat.misteli@gmail.com

Steven Poulakos
Disney Research
steven.poulakos@disneyresearch.com

Mubbasir Kapadia
Rutgers University
mubbasir.kapadia@rutgers.edu

Robert W. Sumner
Disney Research, ETH Zurich
sumner@disneyresearch.com

ABSTRACT

This paper presents a system to experience emergent play within a mixed reality environment. Real and virtual objects share a unified representation to allow joint interactions. These objects may optionally contain an internal mental model to act autonomously based on their beliefs about the world. The experience utilizes intuitive interaction patterns using voice, hand gestures and real object manipulation. We author experience by specifying dependency graphs and behavior models, which are extended to support player interactions. Feedback is provided to ensure the system and player share a common play experience, including awareness of obstacles and potential solutions. An author can mix features from game, story and agent-based experiences. We demonstrate our system through an example adventure game using the Microsoft HoloLens.

KEYWORDS

Mixed Reality, Emergent Play, Interactive Narrative, Cognitive Models, Behavior Trees, Dependency Graphs, Tangible User Interface

1 INTRODUCTION

We propose a system to enable emergent play in a mixed reality environment with real and virtual objects. The play experience may include a mixture of gameplay, narrative and agent-based interactions. The player may creatively interact with the system to progress the experience. We utilize the Microsoft HoloLens as the mixed reality interface. It provides the spatial mapping to place virtual objects within the real physical environment. We additionally detect real objects in the environment and integrate properties of real objects within the play experience.

Our paper makes the following main contributions to realize this vision: (1) We have developed a common representation for both real and virtual objects and a runtime instantiation strategy so they may participate in the experience. (2) These objects may optionally contain an internal mental model to act autonomously and make decisions based on their beliefs about the world. (3) We describe a formalism for defining agent behavior, both in terms of the execution of behavior and the visualization of required logical dependencies. (4) We utilize techniques based on Interactive Behavior Trees [14, 17] for accommodating natural player interaction with real and virtual objects. This includes the use of feedback to ensure the system and player share a common play experience, including awareness of current obstacles and potential solutions. (5) Our system allows for the design of mixed reality experiences, which include characteristics of game, story and agent-based simulation.

We demonstrate our system through an example experience depicted in Fig. 4, which presents an adventure game where the player must overcome obstacles in order to achieve a goal. Obstacles must be overcome by player interactions (e.g. using gesture, voice and real object manipulation) to advance the experience. Our system supports emergent play by giving the player feedback about the current experience and the freedom for mixed reality interactions.

2 RELATED WORK

Salen and Zimmerman broadly define play as free movement within a more rigid structure [28]. The player drives the experience, for example with the goal of "getting to know the properties of objects" or exploration of what can I do with this object" [9]. Our aim is to support emergent play in the context of gameplay, narrative and agent interaction in mixed-reality environments.

Emergent gameplay refers to complex situations that emerge from the interaction of relatively simple game mechanics. For example, the game *Scribblenauts* [11] allows a player to spawn any desired object to solve a puzzle. This is an example of intentional emergence designed into the game. Another example of intentional emergence can be found in games like *Minecraft* [5, 23] where goals can be achieved in multiple ways. Unintentional emergence may happen when a player identifies an unintended use of a game feature. We aim to provide an environment where emergence can take place.

Emergent Narrative may include narrative generation techniques that are responsive to author input or user interaction. Our work is inspired by the linear *narrative generation* system, CANVAS, which enables computer-assisted authoring of 3D animated stories [15]. We adopt aspects of the knowledge representation, animation control and visualization to achieve our immersive play experience. This work also motivates our hybrid representation in which character behavior is defined by an external narrative script or by an internal reasoning process [16].

Interactive narratives afford the player to alter the direction or outcome of a storyline and offer a wide spectrum of experiences [25]. Traditional choose your own adventure experiences offer a strong story with manually-specified author intent. The opposite end of this spectrum includes emergent narrative systems in which characters in the story have strong autonomy and the authored intent is automatically generated [1]. Games like *The Sims* [22] or *Minecraft* [23] do not have a predefined story but rather allow the player to create their own goals and let a story emerge from the possible interactions. Furthermore interactive narrative experiences such as Mimesis [31] and Merchant of Venice [24] offer strong story control

of virtual characters while simultaneously supporting automatically generated stories. Narrative experiences such as Façade [21] are hybrid systems in which virtual characters in Façade exhibit a balance of autonomy and story influence. Our system offers a hybrid of pre-defined stories and an emergent narrative experience for the player.

Emergent agent interactions require a notion of intelligent agents and has been studied in cognitive science to explain the structure of the human mind in a comprehensive computer model [27]. The incorporation of intelligent agents allows a system to utilize autonomous entities that observe the world and act according to their own rationale. They react to changes in the world. One of the main four classes of agents described by Weiss [29] is the Belief-Desire-Intention model where each component is represented as a data structure and is manipulated by the agents and its surroundings. We also draw inspiration from Bratman's theory of human practical reasoning [4] to implement a Belief-Desire-Intention (BDI) software architecture. Applying this model we create decision-theoretic goal-based agents, which have beliefs about itself and other agents, desires and possible actions to follow these desires.

Mixed-reality interaction techniques expand the possible user interfaces where real world objects can provide an intuitive way to interact with virtual content [32]. Intuitive manipulation and interaction of physical objects that provide an interface to the system is known as a Tangible AR interface [3] and is used in examples such as a table top environment [18] or a "MirageTable" [2]. A challenge of Tangible AR interfaces is to show users how to manipulate the real object to achieve the desired command. Providing visual hints [30] can aid this process. We make use of real toys to represent entities of the system and use their position and orientation to provide input to the system. In addition to mixed-reality techniques, we also support hand gestures [19, 20] and voice input to provide a natural human interface.

3 OVERVIEW

Our goal is to create an emergent play experience by mixing real and virtual objects. The objects have a common representation and optionally an internal agent mental model which are both compatible with our interaction design. The interactions are designed with graphical representations. Visual and audible feedback during play ensures the system and player share a common play experience.

Mixed-Reality Smart Objects. We apply the concept of smart objects [13] to represent both real and virtual objects where smart objects have states and affordances, which offer capabilities to interact with other smart objects. Virtual objects may be wished into existence and real objects may be introduced during game-play with the system reacting to them. Smart objects can be redefined during runtime by substituting the smart object script.

Intelligent Agent Architecture. In addition to specifying the states and affordances associated with a smart object, we may define an internal agent model of the smart object. We choose a model which embodies mental attitudes and model them with states and affordances. The model operates on a shared state space and determines both when and which internal character behavior to execute. We use dependency graphs to visualize the model and behavior trees to implement the model.

Mixed-Reality Interaction Design. We support three user interaction types: gesture, physical object manipulation and voice commands. We use formalisms to specify these interactions and their influence on the experience progression. We separate different experience characteristics into game, story or agent simulation.

4 MIXED REALITY SMART OBJECTS

In Mixed Reality (MR), we represent all entities (both real and virtual) as smart objects [13]. A smart object contains a set of states and affordances (or capabilities) to interact. Having the same internal representation of all real and virtual objects provides a consistent specification and interaction possibilities of all objects in the world.

State. The states define the nature of a smart object such as "being asleep" or "level of friendliness". They can be represented by any type (e.g. bool, enum, int). They are part of the shared state-space of the system and can be accessed from anywhere in the system. Since physical objects are also declared as smart objects they also include states.

Affordance. Affordances define the interaction possibilities offered by a smart object and can be invoked by the player, the smart object itself or other smart objects. An example affordance of a smart object could be to speak a defined string or navigate to a certain position. Affordances are specified as execution nodes in a Behavior Tree (BT) [6]. This allows handling of affordance results (success or failure). Affordances can make changes to the shared state-space.

Instantiation. The states and affordances of a smart object are defined in a smart object script. Smart object scripts are defined in a hierarchical manner to support code reuse. Each smart object will always inherit all the states and affordances of its predecessors. The system instantiates a smart object by attaching a smart object script to either a virtual or a tracked physical object. That way the system grants interaction capabilities over the smart objects. The hierarchy allows authoring of events (collection of affordances) that require a certain subtype of smart object rather than specifying a specific smart object. We support two types of instantiation, namely pre-defined and dynamic.

An author may *pre-define* objects by attaching the desired smart object script to it and configuring the initial states. This will allow a player to spawn new objects into the world that are known to the system. For example the player could speak "create a key" and the system will instantiate the pre-defined object "key" as shown in Fig. 4h. Given the hierarchical structure the player also has the ability to introduce new undefined *dynamic* smart objects at runtime by either physical or virtual means. The player can wish for a virtual object which will result in a token being spawned with a google image search to texture the desired object shown in Fig. 4d. After the player introduced a new object and defined it as a certain type of smart object the system attaches the corresponding smart object script and the object will inherit all affordances and states of the specified type. Since scripts can be removed and added

during runtime the player is also able to reconfigure already existing objects by redefining the smart object script.

5 INTELLIGENT AGENT ARCHITECTURE

We utilize a Belief-Desire-Intention (BDI) based agent model [29] to allow real and virtual smart objects to act autonomously and make decisions based on their emotions and the state of the system.

5.1 Belief-Desire-Intention Model

Agent intelligence within the BDI model is specified with three components: belief, desire and intention.

Belief. The belief states represent what the agent thinks of the system. In a trivial case this will be the ground truth of the current system, such as number of objects of a certain type in the world. However, an agent's belief must not necessarily reflect the actual state of the system. It can be mislead by for example only letting the agent observe part of the system and furthermore include emotional beliefs (e.g. as the belief that another agent is friendly) which cannot be trivially measured. A belief is represented as a state in a smart object and is thus part of the shared state-space. It can be of any type (e.g. bool, enum, int). Beliefs of a smart object are influenced by changes of other states in the shared state-space. A belief state is defined internally to the smart object itself.

Desire. The desire states represent the goals of the agent, i.e. the desire to do something such as interacting with an object or running away from a certain area. As the aforementioned belief state a desire is represented as a state in a smart object script. It can be of any type and cannot directly be altered by any other means than the smart object itself. Desires can be manipulated by changes of other states in the shared state-space.

Intention. The intentions are the resulting actions the agent will take given a defined combination of beliefs and desires. They are represented as affordance calls in the system.

5.2 Behavior Tree Representation of Belief-Desire-Intention Model

A BDI at its core is one or more intentions that are triggered by a defined set of beliefs and desires. Beliefs and desires are in turn affected by changes in the shared state-space. These dependencies can be represented in a dependency graph. The concept of dependency graphs have been used commercially in games such as DeathSpank by Electronic Arts [8]. They show all state dependencies of an experience in the form of a polytree. Parents of every node are the preconditions that must be met in order for the the node itself to become true. This allows to author complex experiences better than with a complex story graph where possible path flows would have to be considered. It also aids the implementation of behavior trees to perform routine checks on the changeable states.

Fig. 1 shows how a dependency graph can be used to visualize how desires and beliefs influence intentions and are influenced by the shared state-space. In the directed graph all parents of a node must be fulfilled in order for the node to also be true or executed.

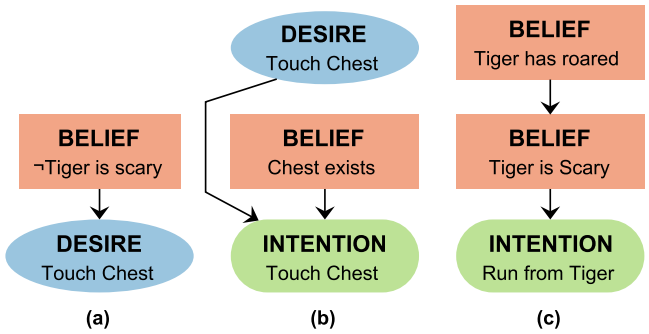


Figure 1: Inter-influence of belief, desire and intention. (a) The belief that the tiger is not scary results in the desire to touch the chest. (b) The desire to touch the chest and the belief that the chest exists results in the agent intention to touch the chest. (c) The belief that the tiger has roared results in the belief that the tiger is scary which results in the agent intention to flee from the tiger.

Behavior Trees (BTs) [6, 12] are represented as hierarchical nodes that control the flow of decision making of an AI entity. It consists of nodes that are either the root, a control flow node or an execution node. A BT is executed by sending a tick with a certain frequency to the root node which forwards this tick to the execution node defined by the control flow nodes. An execution node can either return *running*, which will cause it to consume the next tick, *success* or *failure*. Depending on the finish state (success or failure) the control flow nodes send the next tick to a different execution node. BTs can grow as complex as defined by the author.

The BT model allows sequential routine checking of all defined states. Each state that can be influenced has a routine check of its preconditions. Every node with one or more parent shown in Fig. 1 requires one routine check which is implemented as a sequence in a BT. An example of such a mapping is shown in Fig. 2 with the corresponding dependency graph depicted in Fig. 1b. It is implemented as a sequence in a BT with the first two execution nodes being leaf-asserts validating whether the desire to touch the chest and the belief the chest exists are active. Should either of those asserts fail the sequence will terminate, invalidate the intention and restart. Should both of them succeed the last node, the touch chest intention is executed. Intentions are implemented as an affordance call of the smart object; in this case the touch-chest affordance.

Each smart object contains its own BT which is ticked independently of any other BT. This allows introduction and removal of smart objects with a BDI model during run-time. Discussion of player interaction with the BT-based BDI can be found in Sec. 6.2.

6 MIXED-REALITY INTERACTION DESIGN

The goal of our system is to enable interactions between the mixed-reality smart objects and the player. In this section, we describe how the system considers player interactions in the context of different play experiences.

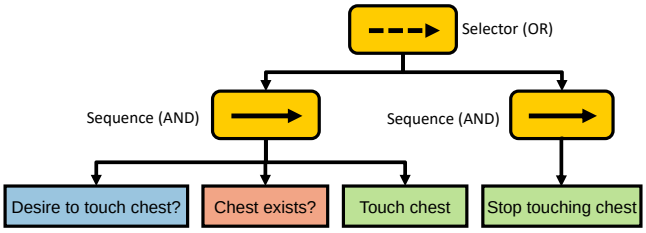


Figure 2: Behavior Tree of Belief-Desire-Intention module shown in Fig. 1b involving the chicken and the chest

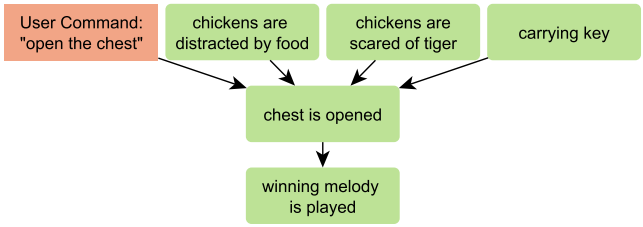


Figure 3: Main dependency subtree showing player interaction-state (red) and system states (green) as preconditions for the system state "chest is opened"

6.1 Supported Player Interactions

The Microsoft HoloLens offers several forms of player interaction to enable a natural play experience. It does not require any controllers leaving the hands of the user free to manipulate physical objects and allows hand gestures. The interaction in our system should be intuitive, require no prior knowledge or experience with MR and no additional hardware setup. We thus chose 3 types of user interaction possibilities:

Gesture Interaction. We included the standard air-tap gesture defined and provided by the HoloLens. It can be configured to, for example, select objects and move them if the corresponding smart object allows it.

Real Object Interaction. We extended the system with the Vuforia library [10] to allow real objects such as children's toys to be tracked and their position and orientation to be available for the system to use. Manipulating its position or orientation can directly influence states of the physical smart object and thus influence the shared state-space. For example a toy animal's rotation can determine its state "is sleeping" by lying on its side.

Voice Interaction. The system was further extended by adding SRGS grammar definition [7]. This allows to author the same command with different variables. For example the commands "*show me the* ⟨*obj*⟩" and "*go to the* ⟨*obj*⟩" work in conjunction with the variable definition ⟨*obj*⟩ = *"dog", "cat", "ball", "user"*. The author can then introduce new objects by only extending the ⟨obj⟩ list without having to add all possible voice command combinations involving the object.

6.2 Interactive Behavior Trees

We utilize the Interactive Behavior Trees (IBT) concept of Kapadia et al. [14] to extend the notion of Behavior Trees (BT) [6, 12] to enable user interaction with the system. Our IBTs were implemented using a library [26] that enables concurrent access to a globally shared state-space by means of an exposed parameter interface. We generate the following three main distinct subtrees to enable interactions with the system: (1) An event-tree that contains all the events (a collection of smart object affordances) which will get triggered as soon as the corresponding event-state is activated. (2) An interaction-tree which enables an interaction-state when a corresponding user interaction mentioned in Sec. 6.1 was registered. (3) A state-monitor-tree that registers the interaction states, checks authored preconditions and enables event states. This state-monitoring-tree or main dependency tree can be visualized with dependency graphs and implemented in a similar way as the BDI models in Sec. 5.

Fig. 3 shows part of the main dependency graph used in our resulting experience. Given a player interaction (red) we can anticipate their intention and provide feedback if a precondition is not yet met. The behavior tree mapping is done the same way as shown in Sec. 5.2 with one main difference. If a leading assert (i.e. one of the preconditions) fail we do not invalidate the intention but let the player know why a precondition failed and possibly give a hint on how to overcome the obstacle.

6.3 Experience characteristics

With our system we can author stories, games, agent-based simulations or any mixture as desired. The characteristics of having a pure game, story or agent simulation are described as follows. A pure **game** experience will not progress without user interaction. The aim is to solve a puzzle or overcome an obstacle. A pure **story** experience will progress without player interaction. A player is able to spectate what is taking place without having the ability to influence the action. A pure **agent-based simulation** experience is also unaffected by any user interaction, but unlike a story the logical decisions of the smart objects and the calling of affordances are not implemented on a global level but on agent-level in form of a BDI for each smart object as described in Sec. 5.

Our concept allows authoring of a seamless hybrid of all three experiences. The main dependency tree gives complete control to the author to create any dependencies. A game requires player interaction to achieve goals. Thus when designing a dependency graph for a game a proportionally large amount of interaction-states are assigned as a precondition to other states. This causes the progression of the experience to be in the players hands and the experience is strongly game-based. Reducing the amount of interaction-states assigned as preconditions in the dependency graph design will make the experience progress more on its own. Events can be authored to invoke more affordances which can lead to a single interaction having a large affect on the progression of the experience. In addition we can define more states being changed in the shared state-space after a single interaction. This can in return result in more preconditions without interaction-state to succeed and further progress the experience autonomously. Designing the experience this way will shift the experience to be more story-based

and is a more passive experience for the player when compared to a game-based experience.

The addition of smart objects that include a BDI-based agent model is independent of the previous design choices. An author can choose to have the intentions of a smart object with a BDI affect the shared state-space which can make the smart objects part of the game- or story-experience. The author may also choose to design a dependency graph which ignores any states that are influenced by the BDI model of a smart object. In that case the smart object will appear to be a background character enhancing the world without affecting it.

To ensure a smart object that includes a BDI will not override or ignore a command from the main dependency tree the concept of a "freedom-belief" is introduced. This belief-state is a precondition for all intentions in the BDI model. If the main dependency tree requires control over the smart object it can falsify the freedom-belief. This will instantly disable all preconditions inside the BDI model and allows the main dependency tree to call any affordance.

7 RESULTS

We demonstrate a seamless merge of all (game, story and agent simulation) aspects of our system in an adventure game where the user must open the chest by manipulating both real and virtual objects. The resulting experience is shown in a storyboard in Fig. 4. Fig. 4a and Fig. 4b show the real and virtual world setup when initializing the experience. After a player uses a voice command to open the chest, a longer event is triggered (Fig. 4c) where the protagonist walks to the chest then stops because chickens were spawned and are flocking to the chest causing him to be scared. The creation of a dynamic object "corn" (Fig. 4d) and definition as food for chickens alters the belief of a subset of chickens that there is chicken food in the world. This will affect their BDI and cause them to run for the introduced object (Fig. 4e). Commanding to open the chest will result in having the next hint provided that the tiger could be woken up. Fig. 4f shows that the tiger is detected (orange pin) and is in a sleeping state (visual "zzZ" and audible purring). Changing its orientation (Fig. 4g) will result in the tiger waking up. This will change the belief of the other subset of chickens, causing them to flee from the tiger. Fig. 4h shows the player having created a predefined object "key" which does not need to be defined by the player. Fig. 4i shows the successful completion of the last dependency by opening the chest.

8 LIMITATIONS AND FUTURE WORK

While we propose design concepts, we do not have a graphical interface to facilitate the design process from conception to implementation. Future work could include mapping of the IBT to a dependency tree visualization. With a provided GUI to design dependency graphs the author can be aided with detection of unreachable states. Further on the dependency graph could then be exported to multiple routine checks represented in an IBT.

The system currently explores the design concepts, but does not employ offline or online planning associated with IBT-based systems. Future work can additionally incorporate planning to generate play experiences based on user interaction.

Future work can also include a completely dynamic voice detection without having to be limited to a predefined grammar. This would increase the possibilities of the user experience and speed up development since no expected words would have to be pre-defined.

The creation of a dynamic object results in a questionnaire that always starts at the top of the smart object hierarchy. The addition of assumptions based on an obtained knowledge-base could speed up or even replace the process of defining an unknown object.

9 CONCLUSION

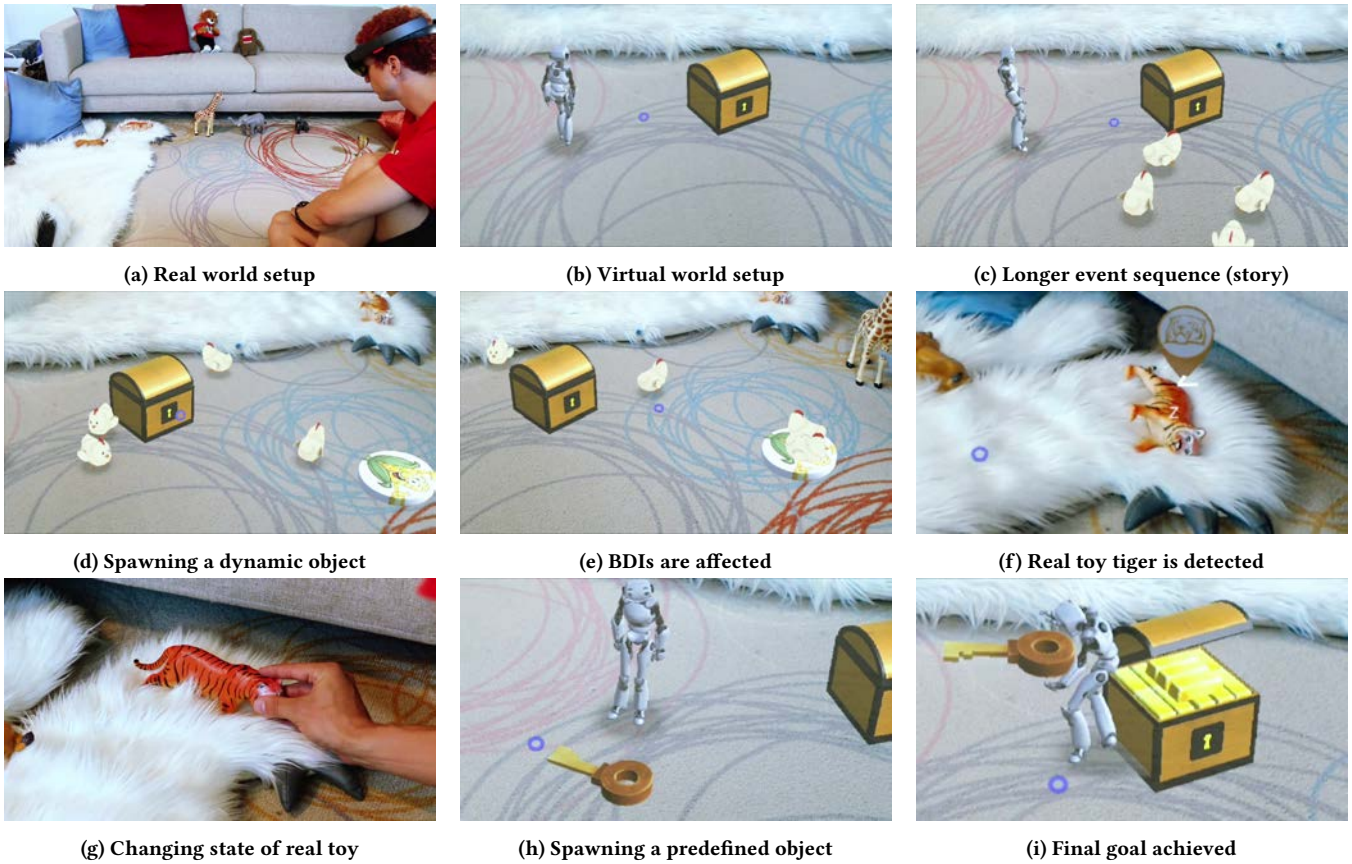
We demonstrated an emergent play experience in a mixed reality environment using the Microsoft HoloLens. A common representation has been applied to both real and virtual objects allowing interaction within the play experience. The player has the option to introduce new objects which are either pre-defined or defined by the player at runtime. These objects optionally have an internal mental model to autonomously respond to the environment. We described an approach for representing complex system behavior using a combination of dependency graph and behavior trees. We additionally utilized techniques based on interactive behavior trees for accommodating natural player interactions with real and virtual objects. This includes the specification of player feedback to ensure the system and the player share a common play experience. Our interaction design allows for game, story and agent-based play experiences.

ACKNOWLEDGEMENT

The authors would like to thank Maurizio Nitti, Alessia Marra and Mattia Ryffel for providing support with the virtual game assets. The cartoon corn image in Figure 4 (d-e) is provided by ©memoangeles - Can Stock Photo Inc.

REFERENCES

[1] Ruth Aylett. 1999. Narrative in virtual environments-towards emergent narrative. In *Proceedings of the AAAI fall symposium on narrative intelligence*. 83–86.
[2] Hrvoje Benko, Ricardo Jota, and Andrew Wilson. 2012. MirageTable: freehand interaction on a projected augmented reality tabletop. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 199–208.
[3] Mark Billinghurst, Hirokazu Kato, and Ivan Poupyrev. 2008. Tangible augmented reality. *ACM SIGGRAPH ASIA Courses* 7 (2008).
[4] M. Bratman. 1987. *Intention, plans, and practical reason*. Harvard University Press.
[5] Josh Bycer. 2015. Examining Emergent Gameplay. http://www.gamasutra.com/blogs/JoshBycer/20150916/253682/Examining_Emergent_Gameplay.php. (2015).
[6] R. G. Dromey. 2003. From requirements to design: formalizing the key steps. In *First International Conference on Software Engineering and Formal Methods*. 2–11. https://doi.org/10.1109/SEFM.2003.1236202
[7] Max Froumentin. 2005. The W3C Speech Interface Framework Media Types: application/voicexml+ xml, application/ssml+ xml, application/srgs, application/srgs+ xml, application/ccxml+ xml, and application/pls+ xml. (2005).
[8] Ron Gilbert. 2014. Puzzle Dependency Charts. http://grumpygamer.com/puzzle_dependency_charts. (2014).
[9] Corinne Hutt. 1966. Exploration and play in children. In *Symposia of the Zoological Society of London*, Vol. 18. London, 61–81.
[10] PTC Inc. 2014. Vuforia Augmented Reality SDK. (May 2014). https://www.vuforia.com
[11] Warner Bros. Entertainment Inc. 2013. Scribblenauts. (2013). https://www.scribblenauts.com
[12] D Isla. 2005. GDC 2005 Proceeding: Handling Complexity in the Halo 2 AI. (2005). Retrieved August 31, 2017 from https://www.gamasutra.com/view/feature/130663/gdc_2005_proceeding_handling_.php
[13] Marcelo Kallmann and Daniel Thalmann. 1999. Direct 3D Interaction with Smart Objects. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST '99)*. ACM, New York, NY, USA, 124–130. https://doi.org/10.1145/323663.323683



(a) Real world setup (b) Virtual world setup (c) Longer event sequence (story)

(d) Spawning a dynamic object (e) BDIs are affected (f) Real toy tiger is detected

(g) Changing state of real toy (h) Spawning a predefined object (i) Final goal achieved

Figure 4: Storyboard of experience showing different aspects of the system's capabilities

[14] Mubbasir Kapadia, Jessica Falk, Fabio Zünd, Marcel Marti, Robert W Sumner, and Markus Gross. 2015. Computer-assisted authoring of interactive narratives. In *Proc. of the 19th Symposium on Interactive 3D Graphics and Games*. ACM, 85–92.

[15] Mubbasir Kapadia, Seth Frey, Alexander Shoulson, Robert W. Sumner, and Markus Gross. 2016. CANVAS: Computer-assisted Narrative Animation Synthesis. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '16)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 199–209. http://dl.acm.org/citation.cfm?id=2982818.2982846

[16] Mubbasir Kapadia, Alexander Shoulson, Cyril Steimer, Samuel Oberholzer, Robert W. Sumner, and Markus Gross. 2016. An Event-centric Approach to Authoring Stories in Crowds. In *Proceedings of the 9th International Conference on Motion in Games (MIG '16)*. ACM, New York, NY, USA, 15–24. https://doi.org/10.1145/2994258.2994265

[17] Mubbasir Kapadia, Fabio Zünd, Jessica Falk, Marcel Marti, Robert W. Sumner, and Markus Gross. 2015. Evaluating the Authoring Complexity of Interactive Narratives with Interactive Behaviour Trees. In *Foundations of Digital Games (FDG'15)*. 9.

[18] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, and K. Tachibana. 2000. Virtual object manipulation on a table-top AR environment. In *Proceedings IEEE and ACM International Symposium on Augmented Reality (ISAR 2000)*. 111–119. https://doi.org/10.1109/ISAR.2000.880934

[19] Hui Liang, Junsong Yuan, Daniel Thalmann, and Nadia Magnenat Thalmann. 2015. AR in Hand: Egocentric Palm Pose Tracking and Gesture Recognition for Augmented Reality Applications. In *Proceedings of the 23rd ACM International Conference on Multimedia (MM '15)*. ACM, New York, NY, USA, 743–744. https://doi.org/10.1145/2733373.2807972

[20] S. Malik, C. McDonald, and G. Roth. 2002. Hand tracking for interactive pattern-based augmented reality. In *Proceedings. International Symposium on Mixed and Augmented Reality*. 117–126. https://doi.org/10.1109/ISMAR.2002.1115080

[21] Michael Mateas and Andrew Stern. 2003. Façade: An experiment in building a fully-realized interactive drama. In *Game developers conference*, Vol. 2.

[22] Maxis. 2000. The Sims. [CD-ROM]. (2000).

[23] Mojang. 2011. Minecraft. [CD-ROM]. (2011).

[24] Julie Porteous, Marc Cavazza, and Fred Charles. 2010. Applying Planning to Interactive Storytelling: Narrative Control Using State Constraints. *ACM Trans. Intell. Syst. Technol.* 1, 2, Article 10 (Dec. 2010), 21 pages. https://doi.org/10.1145/1869397.1869399

[25] Mark O. Riedl and Vadim Bulitko. 2013. Interactive Narrative: An Intelligent Systems Approach. *AI Magazine* 34, 1 (2013), 67–77.

[26] Alexander Shoulson, Francisco M. Garcia, Matthew Jones, Robert Mead, and Norman I. Badler. 2011. Parameterizing Behavior Trees. In *Proceedings of the 4th International Conference on Motion in Games (MIG'11)*. Springer-Verlag, Berlin, Heidelberg, 144–155. https://doi.org/10.1007/978-3-642-25090-3_13

[27] John Sweller, Jeroen J. G. van Merrienboer, and Fred G. W. C. Paas. 1998. Cognitive Architecture and Instructional Design. *Educational Psychology Review* 10, 3 (01 Sep 1998), 251–296. https://doi.org/10.1023/A:1022193728205

[28] Katie Salen Tekinbaş and Eric Zimmerman. 2003. *Rules of play: game design fundamentals*. MIT Press, Cambridge, Mass.

[29] Gerhard Weiss (Ed.). 2013. *Multiagent systems* (second edition ed.). The MIT Press, Cambridge, Massachusetts.

[30] Sean White, Levi Lister, and Steven Feiner. 2007. Visual hints for tangible gestures in augmented reality. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE, 47–50.

[31] R Michael Young, Mark O Riedl, Mark Branly, Arnav Jhala, RJ Martin, and CJ Saretto. 2004. An architecture for integrating plan-based behavior generation with interactive game environments. *Journal of Game Development* 1, 1 (2004), 51–70.

[32] Feng Zhou, Henry Been-Lirn Duh, and Mark Billinghurst. 2008. Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE Computer Society, 193–202.