Toward Generating 3D Games with the Help of Commonsense Knowledge and the Crowd

Rania Hodhod[1], Marc Huet[2], and Mark Riedl[2]
[1]TSYS School of Computer Science, Columbus State University
[2]School of Interactive Computing, Georgia Institute of Technology
hodhod_rania@columbusstate.edu; {mbhuet, riedl}@gatech.edu

Abstract

Procedural game generation is the automatic creation of all aspects of a playable computer game. Procedural game generation systems require specialized knowledge, virtual worlds, and art assets. In this paper, we show how 3D graphical scenes for interactive fictions can be automatically generated with only knowledge that is readily available in existing knowledge bases or can be acquired via crowdsourcing. The key to 3D scene generation is commonly accepted spatial relationships between different types of objects in different types of scenes. We use a crowdsourcing game to automatically and rapidly acquire spatial relations. The spatial relations are used by an intelligent scene generation system that selects and configures 3D assets within a virtual geometric space.

Introduction

Procedural content generation is the automatic creation of video game content, where content means anything that is traditionally created by an artist or a designer, such as maps, textures, levels, and objects. Procedural game generation is the automatic and simultaneous creation of all aspects of a playable computer game experience (Togelius et al. 2011; Hendrikx et al. 2013). Procedural game generation is task that requires a large amount of outside information and knowledge (Nelson & Mateas 2008; Smith and Mateas 2010) Treanor et al. 2012; Cook et al. 2013; Cook, and Colton 2014). Often a procedural game generation system has access to a virtual world, art assets such as sprites or 3D models, design criteria, and evaluation functions. The Scheherazade system (Li, Lee-Urban, and Riedl 2012; Li et al. 2013) demonstrated that a particular type of game—the interactive fiction, a text-based game in which one issues commands to progress through a narrative—can be generated without any knowledge except for that that can be accessed through existing online knowledge bases and corpora or can be automatically, quickly, and easily crowdsourced. However, due to emphasis on learning plot points from crowdsourcd textual corpora, the Scheherazade system only generates textual adventure games.

In this paper we present work exploring the question of whether a *graphical* interactive narrative can be automatically generated. We extend the existing work by Li, Lee-Urban and Riedl with a procedural 3D graphical scene generation technique that, true to the prior motivation, does not use any knowledge that cannot be acquired through existing knowledge bases or crowdsourced directly from humans in a quick and automated fashion. *Procedural scene generation* is a process of automatically rendering a 3D graphical environment portraying a real world environment. For example, one may request a scene be generated for a "restaurant". Such a system should be able to respond to any conceivable request.

Procedural scene generation is related to text-to-scene generation, but instead of taking a textual description of the scene, the process relies on commonsense knowledge about what typical scenes of the given type are expected to look like. The key to procedural scene generation is commonsense knowledge. We humans, living in the real world, have familiarity with a wide range of environments and have expectations about what one finds in these environments and how they are arranged. Commonsense knowledge bases such as CYC (Lenat 1995) and ConceptNet (Havasi, Speer, and Alonso 2007) are sources that can provide us with lists of objects that might exist in the environment of interest. However, what is missing from all commonsense knowledge bases to date is spatial information: how objects in an environment tend to be situated with respect to each other. For example, existing com-

monsense knowledge bases can tell us that tables, chairs, and doors are commonly found in restaurants, but not that chairs are typically found next to tables and far from doors.

One possible solution to the problem of missing spatial relational commonsense knowledge is to handcraft a custom knowledgebase for a large number of possible scenes. However, it is not clear how this strategy would be different from simply building the 3D scenes by hand. Further, there would always be scenes that could not be generated. Another possible solution is to crowdsource the requisite spatial knowledge in a just-in-time fashion. Crowdsourcing is the outsourcing of complicated tasks to a number of anonymous human individuals and then aggregating the results (Quinn and Bederson 2011). Crowdsourcing is a cheap and effective alternative for populating commonsense knowledge bases (Havasi, Speer, and Alonso 2007; Siorpaes and Hepp 2008).

In this paper, we extend Scheherazade with a procedural scene generation technique that uses just-in-time crowdsourcing to acquire the necessary spatial relation information for a scene. The spatial relations are acquired via a crowdsourcing computer game (game with a purpose) in which points are earned by matching spatial relations to pairs of objects that may appear in a particular scene. The spatial relations are used by an intelligent scene generation system that selects and configures 3D assets within a virtual geometric space. To our knowledge, this is the first consideration of procedural scene generation for games that does not assume a pre-existing knowledgebase. It is also the first example of how playing a computer game can help create a computer game.

Background and Related Work

Computer game generation is a nascent area of Game AI research. A number of researchers have developed intelligent systems that produce playable games in a single genre (Nelson & Mateas 2008; Smith and Mateas 2010) Treanor et al. 2012; Cook et al. 2013; Cook, Colton, and Gow 2014) or across many game genres (Zook and Riedl, 2014). Most game generation systems focus on 2D games. The Angelina system is notable in that it has been updated to produce 3D maze games (Cook and Colton 2014).

Procedural content generation has been used to perform scene layout. CAPS (Xu, Stewart, and Fiume 2002) is a constraint-based automatic placement system, which lays out a scene given a set of hand-coded constraints of varying degrees of specificity. The furniture arrangement technique by Yu et al. (2011) searches for a configuration of a priori known types of furniture based on a set of heuristics inspired by interior design.

Procedural scene generation is a type of text-to-scene system. Text-to-scene systems take a number of natural language sentences describing a scene and produce a 3D graphical visualization. Confucius (Ma 2006) is a multimodal text-to-animation system that generates animations of virtual humans from single sentences containing an action verb. In these systems the referenced objects, attributes, and actions are typically relatively small in number or targeted to specific pre-existing domains. WordsEye (Coyne and Sproat 2001) uses natural language input in which verbs are resolved to semantic frames that are then mapped to corresponding poses and spatial relations. WordsEye database includes information about the objects' typical parts, typical location and typical objects nearby them. Zitnick, Parikh, and Vanderwende (2013) learn probabilistic placements of items in 2D scenes by watching humans illustrate text descriptions. Scenes can only be composed from a finite set of known art assets.

The successes of the above systems are functions of the quality of knowledge they are working from. One challenge scene layout and text-to-scene systems face is the addition of new types of objects, which requires not only a new class but also understanding of the relationships with all other types of objects. Crowdsouring seems to provide a solution to this problem through the automated collection of different objects and their spatial relationships. WordsEye (2001) knowledge base is partially crowdsourced. The system by Zitnick, Parikh, and Vanderwende requires a large corpus of example scene layouts, which can be crowdsourced. As with most crowdsourced knowledge bases, no incentive is provided for people to contribute.

Games with a Purpose (GWAPs) (von Ahn and Dabbish 2008) are games in which players generate useful data or solve problems as a by-product of play. For example, players may label images (von Ahn and Dabbish 2004), discover the shapes of proteins (Cooper et al. 2010), or categorize concepts to develop an ontology (Siorpaes and Hepp 2008; Carranza and Krause 2012). von Ahn and Dabbish (2008) give a number of design templates for GWAPs. In particular *output-agreement games* ask two players to collaborate to generate labels for things (e.g., images) without collusion. Players are rewarded when labels match. The ESP Game (von Ahn and Dabbish 2004) is the canonical example of an output-agreement game. Our approach to procedural scene generation uses an output-agreement GWAP to acquire spatial relations for items in a scene. However, instead of asking players to generate text labels, players select from a fixed vocabulary of spatial relations.

The Scheherazade system (Li, Lee-Urban, and Riedl, 2012; Li et al. 2013) attempts to create a novel, fictional narrative about a simple, user-provided topic. For example, the user may request a story about a "bank robbery". The system uses crowdsourcing to rapidly acquire a number of linear narrative examples about typical ways in which the topic might occur. In other words, we collect human experiences in narrative form and learn a generalized model

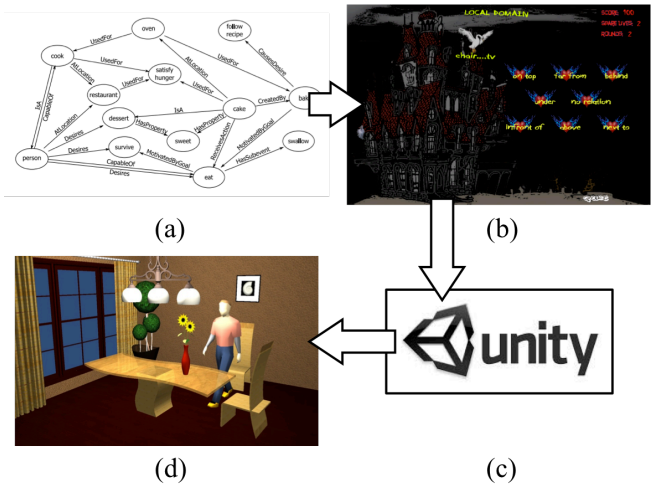

Figure 1. The pipeline for scene generation for a requested game environment.

about the topic domain. The details of how Scheherazade learns topic models and generates stories is beyond the scope of this paper; See Li et al. (2013) for details. Scheherazade was modified to generate playable text-based interactive fictions (Li, Lee-Urban, and Riedl 2012).

Procedural Scene Generation

Our approach to procedural scene generation takes a simple term describing a scene (e.g., "restaurant") and produces a fully rendered 3D visualization of the environment. Procedural scene generation is a process of arranging the items commonly found in an environment to conform to the expectations of the user. However, we do not assume that the system knows anything about the type of environment. Thus, our system must solve four problems. First, it must determine what items should be in the requested scene. A common sense knowledge base structured in the form of a semantic net is used for this purpose, see Figure 1(a). Second, it must determine typical spatial relationships between the items in the scene. A simple game is designed to collect these relations called HeartESP as shown in Figure 1(b). Finally, it must procure graphical art assets for the items in the scene. A 3D models database is used to collect the assets which are then passed to the Unity 3D Game Engine to position the graphical art assets in the virtual environment, see Figure 1(c) and Figure 1(d).

Generating Scene Items

In this section we discuss how our system "expands" the original query of an environment type into a set of items that should populate the scene. Common sense knowledge bases contain a vast amount of general knowledge that people are expected to possess, represented in an ontological form that it is available to artificial intelligence programs. ConceptNet (Havasi, Speer, and Alonso 2007) is a knowledge base presently consisting of over 1.6 million assertions of common sense knowledge encompassing the spatial, physical, social, temporal, and psychological aspects of everyday life. Knowledge in Concept Net is organized in the form of a semantic net in which nodes are used to represent the different concepts and edges are used to describe relations between these concepts. Concepts and their associated relations can be described in the form of production rules, e.g., *can_exist(payment_counter, bank)*. ConceptNet does not contain spatial relations. We utilize ConceptNet to "expand the query" by collating all conceptually related objects that can exist in the given environment type (e.g., "restaurant").

The use of the ConceptNet KB is both a strength and a weakness. ConceptNet is noisy in the sense that it contains concepts that are sometimes non-sensical from the perspective of procedural scene generation. For example, *can_exist (trust, bank)* would result in our system attempting to place "trust" in the scene despite the fact that it is an abstract concept. Consequently, we apply a filtration step that eliminates the concepts that appear to be noise from our purposes. We implement a rule that searches the network for a *used_for* predicate for each candidate concept retrieved. The *used_for* predicate only uses physical objects in the first parameter. For example, used_for*(atm, withdraw_money)* would allow us to keep an ATM (Automated Teller Machine) as an item in a scene about a bank. Despite our satisfaction with the generated list of objects, it is possible for our system to retrieve equivalent concepts with different symbolic names, such as *coin* and *currency*. We were able to account for this later in the pipeline when we acquire spatial relations.

Crowdsourcing Spatial Relations

No commonsense knowledge bases contain common typical spatial relations between objects. The idea behind GWAPs is to use the computational power of humans to perform tasks that computers cannot currently perform by presenting the task as a game (von Ahn and Dabbish 2004; von Ahn and Dabbish 2008). Unlike other approaches to crowdsourcing that pay crowd workers, GWAPs incentivize people to perform work in exchange for entertainment.

HeartESP is an output-agreement game based on The ESP Game. In The ESP Game, two players collaborate to label images while remaining anonymous to each other to prevent collusion; players receive points when labels match. Unlike in The ESP Game, HeartESP players must select spatial relations from a fixed set of relations: on top, far from, behind, under, in front of, above, next to, and no relation. No relation is useful because some objects in a scene are truly unrelated. It is also possible for objects to

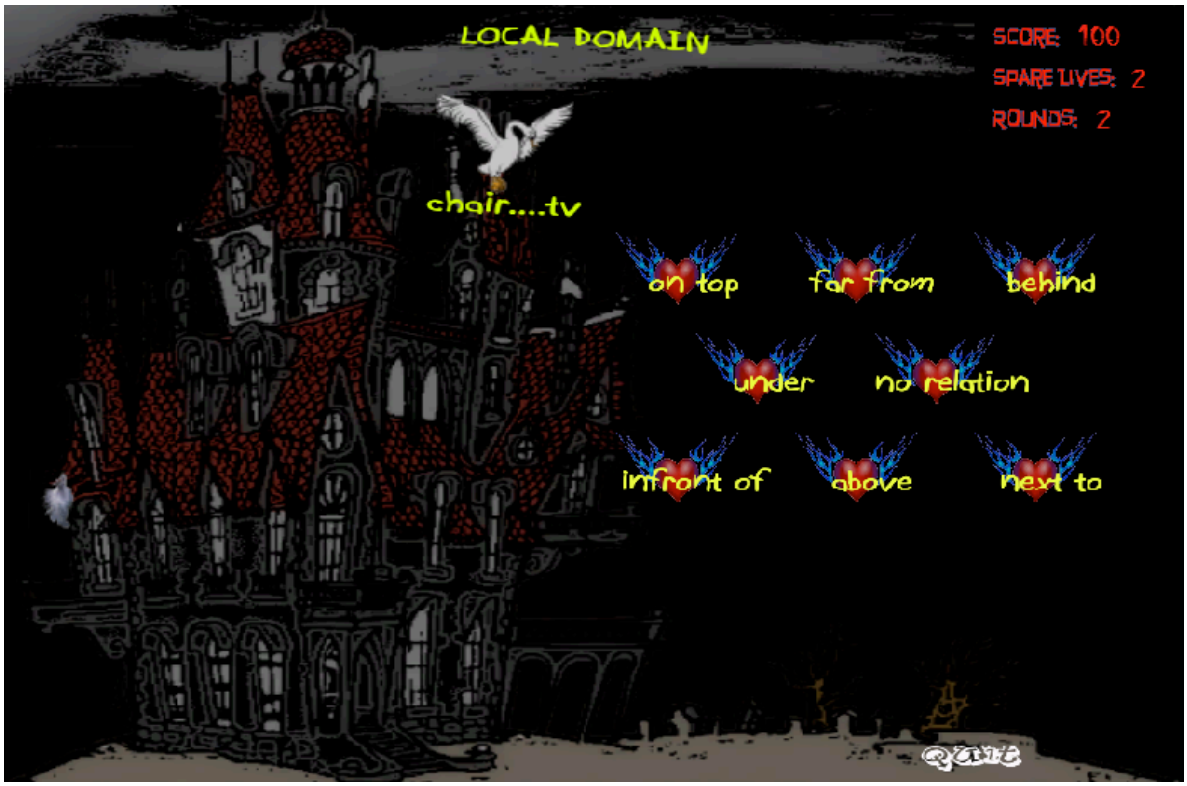

Figure 2. Snapshot the HeartESP game.

be synonymous (e.g., "money" and "currency," in which case players are instructed to also choose "no relation".

When a scene is being procedurally generated for the first time and requires spatial relations, the HeartESP game is launched with a list of items identified for the scene from the earlier part of the pipeline. Additionally, we add three special objects: floor, ceiling, and wall, which help determine object locations relative to the fixed boundaries of the scene. The next sections describe how the game elicits and verifies spatial relations for objects for a scene.

Background Story

HeartESP has a background story to engage the players and frame the rules and visuals of the game. The story is about a witch who turned the queen to a swan. The queen has two twin daughters who need to combine their efforts through reading each others' minds to provide the same answer to the witch's questions. Players assume the role of one of the twin daughters.

Game Mechanics

The game is implemented in Unity and hosted on a website. Once a player loads the game, he or she enters a "waiting room" until at least one other player is available. If another player is not available after a fixed amount of time, the player will be matched against a simulated player that uses play traces of prior players. Pairing a human player with a trace of a previous player is a common strategy used in GWAPs to ensure the game is always playable regardless of the number of people online (c.f., Siorpaes and Hepp 2008; von Ahn and Dabbish 2008; Siu, Zook, and Riedl 2014). Partnered players are never aware of each others' identity and they cannot communicate.

Once players are partnered, they are told the type of scene that they should consider. Each round of play involves a random pair of objects that are carried across the top of the screen by a swan. When the swan reaches the far side of the screen, the round is over. Before the round ends, each player must click on one of a fixed set of spatial relation terms (see Figure 2). Both players will be shown the same pair of objects. The task of each player is to pick the spatial relation between the objects that he or she believes the other player will pick. Visual feedback (a falling feather signifying the breaking of the spell) is shown when players agree and both players receive 100 points. For each round, one player receives an additional 10 points if there is agreement but one player selected the agreed-upon relation more quickly. If the players disagree, they both lose a life. The game ends after 10 rounds or when the players have lost all of their lives.

After each round, a mini game starts that allows the players to increase their scores regardless of the game's evaluation of their choices. In the mini game, players can increase their scores by clicking on randomly appearing coins. The purpose of the mini game is to keep the player motivated and focused on score. Siu, Zook, and Riedl

(2014) found that competing for points increases player engagement in output-agreement games. The presence of mini game aims to engage the player and motivates him during his play.

Reliability of Information

Crowd workers are prone to producing noisy results. This is especially true in computer games with a time limit and in which bonus points are awarded to faster selections. The ESP Game assumes that agreement between two people is a reliable signal for correct data. This assumption only holds when the number of possible labels is extremely large (such as the number of possible words in the English language); the possibility of two matching but incorrect labels is very low. HeartESP has only eight possible labels. Consequently, we do additional statistical confidence testing to ensure that we only add reliable data to our spatial relation knowledge base. If p is probability of a spatial relation being incorrectly chosen by two players simultaneously, then after n repetitions by different pairs of players the probability of corruption is p^n because repetitions are independent of each other. When p^n is less than a small threshold, we assume the spatial relation between two objects is true.

Art Assets

Once a set of objects and their spatial relations has been collected for a particular setting, art assets must be acquired. 3D graphical models were downloaded from the Sketchup Warehouse (https://3dwarehouse.sketchup.com). Sketchup is a 3D modeling software program that uses real-world units of measurement (e.g., meters) so that all models are proportional, facilitating model reuse with the need for manual scaling. Following Cook and Colton (2014), our system queries the Sketchup Warehouse online repository with the name of the item. The first 3D model is picked from the list of retrieved models. If the search query fails, a small cube is used as a placeholder in the 3D scene. Future work is necessary to automatically determine the scale of the placeholder box and automatically download an image to place on the sides of the placeholder box.

Scene Generation

The scene is produced by procedurally placing art assets in a 3D graphical space based on their spatial relations. We currently assume that each scene takes place indoors and thus generate a room—a simple cube with one side missing (so that the inside of the room can be seen). The placement algorithm is based on constraint satisfaction strategies, but specialized for scene generation.

Because gravity is a constant, the scene generator starts with items that are on the floor. All objects that have the *on* relationship with the floor are selected and sorted in a queue based on the total number of relations the object appears in. Prioritizing variables that have the greatest number of constraints is a common optimization strategy in constraint satisfaction algorithms. For each object in the sorted list, the scene generator considers all points on the top of the surface that the object is *on*. Points that do not satisfy all spatial relations with items already placed in the environment are pruned from consideration and one of the remaining points is chosen randomly. Because spatial relations are reciprocal, all spatial relations will have been considered by the time the last item is placed.

It is possible, however, that no point can be identified that satisfies all spatial relations. In this case, one spatial relation is randomly dropped and the point search restarts. Relations are dropped until a point can be identified or until all relations involving the object have been dropped, in which case the object is not placed in the scene.

Once all objects on the floor have been placed, the scene generator moves on to objects that are on the currently placed objects. This process completes until all objects that are on other objects are placed. Objects that are on walls and the ceiling are placed last.

Each generated arrangement is given an incompleteness score. The incompleteness score of an arrangement is computed as the number of relations that were dropped plus an additional penalty for each object that was not placed proportional to its size. Thus, ignoring a large piece of furniture is considered more severe than ignoring a small object such as a fork that is likely to be overlooked by the user. Due to random decisions made in the scene generation process, the arrangement algorithm is run many times and the scene with the lowest incompleteness score is kept as the final scene.

Examples of a generated scene for "kitchen" can be seen in Figure 3. Figure 3(a) has the highest incompleteness score (609.57) while the scene shown in Figure 3(d) has the best score (253.8). Figures 3(b) and 3(c) have intermediate incompleteness scores. These scenes were generated from 18 objects (including wall, floor and ceiling) and 69 relations. Future work is necessary to evaluate the quality of generated scenes as perceived by human users.

Discussion

Procedural game generation promises to substantially reduce the manual effort necessary for producing playable game experiences. Procedural game generation may allow non-programmers and people without game design and development expertise to rapidly bring their game ideas to life. However, procedural game generation systems require substantial amounts of specialized knowledge that is not always available. If specialized knowledge is hard-coded into a procedural game generation system, it limits what games can be produced automatically. For example, many

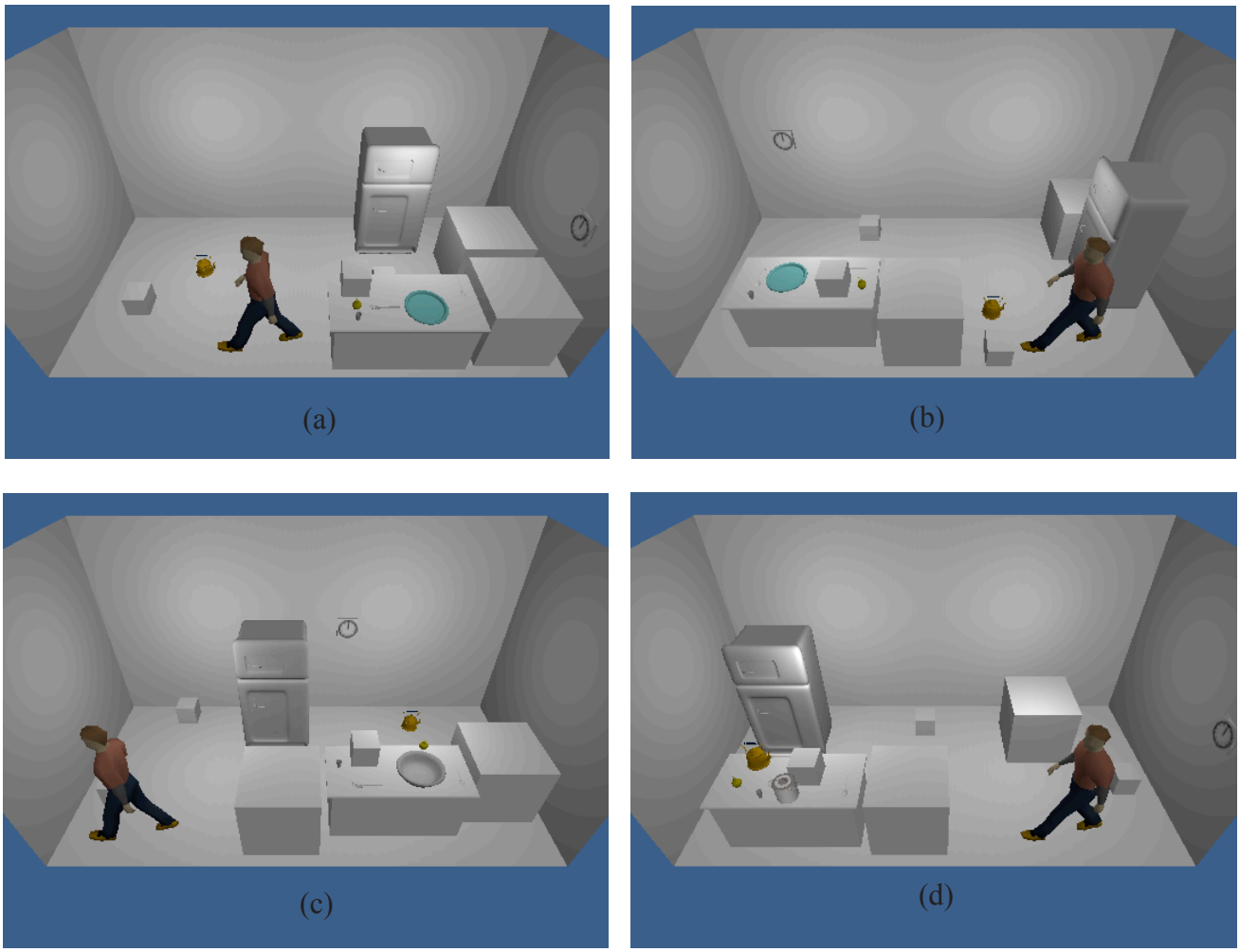

Figure 3. Example scenes automatically generated for a kitchen, in order of decreasing incompleteness score.

procedural game generation systems to date make strong assumptions about the game genre and/or topics.

The Scheherazade system can generate text-based interactive fictions about any conceivable topic by acquiring the knowledge from humans via crowdsourcing. This work extends Scheherazade with procedural 3D graphical scene generation, requiring commonsense knowledge about items that should appear in scenes and commonsense knowledge about spatial relationships between objects in each type of scene. Addressing the question of incentives for human crowd workers, we present a technique for using a game to acquire requisite knowledge. In that sense, humans play computer games in order to generate a computer game.

Future work is required to address a number of aspects of the system that are not entirely complete. As mentioned earlier, we require a more intelligent means of handling the situation wherein 3D art assets are not found on the Internet. Orientation of objects in the scene is not yet addressed. It should be possible to incorporate some of the design heuristics from Yu et al. (2011) to preserve navigability of scenes by animated NPCs.

We have not yet evaluated the accuracy of spatial relations acquired by HeartESP. Because generated scenes appear reasonable, we anticipate that the acquired knowledge is reliable. However, anecdotal evidence suggests that HeartESP is not fun to play for long periods of time; a problem with many GWAPs. While scenes with lower scores visibly have fewer errors, we have also not yet evaluated the system to determine the extent to which scenes are recognizable nor whether our generation technique with multiple restarts is sufficient.

Despite limitations and future work, preliminary results demonstrate it is potentially feasible for an intelligent system to procedurally generate 3D graphical adventure games using only the information available in existing knowledge bases and through crowdsourcing of any specialized knowledge. The concept that playing a game can assist with the production of a game is a compelling vision of how playing and constructing games can be unified under a common framework.

Acknowledgements

We gratefully acknowledge the support of the U.S. Defense Advanced Research Projects Agency (DARPA). Special thanks to Stephen Lee-Urban, John Rafferty, and Matt Lee.

References

Carranza, J., and Krause, M. 2012. Evaluation of game designs for human computation. *Proceedings of the 2012 AAAI Workshop on Human Computation in Digital Games and Artificial Intelligence for Serious Games*.

Cook, M. and Colton, S. 2014. Ludus ex machina: Building a 3D game designer that competes alongside humans. *Proceedings of the 5th International Conference on Computational Creativity*.

Cook, M., Colton, S., Raad, A., Gow, J. 2013. Mechanic Miner: Reflection-driven game mechanic discovery and level design. *Proceedings of the 16th European Conference on the Applications of Evolutionary Computation*.

Cooper, S., Khatib, F., Treuille, A., Barbero, J., Lee, J., Beenen, M., Leaver-Fay, A., Baker, D., Popović, Z., and others. 2010. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307), 756–60.

Coyne, B, and Sproat, R. 2001. WordsEye: An automatic text-to-scene conversion system. *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*.

Havasi, C., Speer, R., and Alonso, J. 2007. ConceptNet: A lexical resource for common sense knowledge. In N. Nicolov, G. Angelova, and R. Mitkov (Eds.) *Recent Advances in Natural Language Processing V: Selected Papers from RANLP 2007*.

Hendrikx, M., Meijer, S. Van Der Velden, J. and Iosup, A. 2013. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications,* 9(1).

Lenat, D. 1995. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM,* 38(11), 33–38.

Li, B. Lee-Urban, S., Johnston, G., and Riedl, M.O. 2013. Story generation with crowdsourced plot graphs. *Proceedings of the 27th AAAI Conference on Artificial Intelligence*.

Li, B., Lee-Urban, S., and Riedl, M.O. Toward autonomous crowd-powered creation of interactive narratives. *Proceedings of the 5th AAAI Workshop on Intelligent Narrative.*

Ma, M. 2006. *Automatic Conversion of Natural Language to 3D Animation*. Ph.D. Thesis, University of Ulster.

Nelson, M. and Mateas, M. 2008. An interactive game-design assistant. *Proceedings of the 2008 International Conference on Intelligent User Interfaces*.

Quinn, A., and Bederson. B. 2011. Human computation: A survey and taxonomy of a growing field. *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*.

Siorpaes, K, and Hepp, M. 2008. Games with a purpose for the semantic web. *IEEE Intelligent Systems,* 23(3), 50–60.

Siu, K., Zook, A., and Riedl, M.O. 2014. Collaboration versus competition: Design and evaluation of mechanics for games with a purpose. *Proceedings of the 9th International Conference on the Foundations of Digital Games*.

Smith, A. and Mateas, M. 2010. Variations Forever: Flexibly generating rulesets from a sculptable design space of mini-games. *Proceedings of the 2010 IEEE Conference on Computational Intelligence in Games*.

Togelius, J., Yannakakis, G., Stanley, K., and Browne, C. 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3), 172–86.

Treanor, M., Schweizer, B., Bogost, I., and Mateas, M. 2012. The micro-rhetorics of Game-O-Matic. *Proceedings of the 2012 International Conference on the Foundations of Digital Games*.

von Ahn, L., and Dabbish, L. 2004. Labeling images with a computer game. *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*.

von Ahn, L., and Dabbish, L. 2008. Designing games with a purpose. *Communications of the ACM*, 51(8), 58–67.

Xu, K., Stewart, J., and Fiume, E. 2002. Constraint-based automatic placement for scene composition. *Proceedings of the 2002 Graphics Interface Conference*.

Yu, L-F, Yeung, S-K, Tang, C-K, Terzopoulos, D., Chan, T.F., and Osher, S.J. 2011. Make it home: Automatic optimization of furniture arrangement. *ACM Transactions on Graphics*, 30(4), Article 86.

Zitnick, L.C., Parikh, D., and Vanderwende, L. 2013. Learning the Visual Interpretation of Sentences. *Proceedings of the 2013 International Conference on Computational Vision*.

Zook, A. and Riedl, M.O. 2014. Automatic game design via mechanic generation. *Proceedings of the 28th AAAI Conference on Artificial Intelligence.*