Towards an AI Holodeck: Generating Virtual Scenes from Sparse Natural Language Input

**Jason Smith, Nazanin Alsadat Tabatabaei Anaraki,
Atefeh Mahdavi Goloujeh, Karan Khosla,
Brian Magerko**

Georgia Institute of Technology
Atlanta, Georgia USA
{jsmith775, nazanin.tbt, atefehmahdavi, kkhosla7}@gatech.edu

Abstract

The *Holodeck*, a virtual reality simulator from the television show Star Trek, is known as the "holy grail" of interactive narrative experiences. However, while there have been approaches to various components of a theoretical Holodeck, scene generation from dialogue is often overlooked. This paper introduces a prototype AI Holodeck application for scene generation, demonstrating the use of Natural Language Processing and a corpus of spatial data. The application creates scenes from user input text and fills those scenes with objects and relationships not explicitly defined by the user. This paper discusses potential use cases of scene generation in creating environments for interactive narrative, virtual reality, and other development opportunities.

1 Introduction

The Holodeck is a fictional virtual reality device in the television show Star Trek, taking the form of a blank room that generates interactive characters and objects dictated by voice commands from people inside of it. It represents a "holy grail" of interactive virtual reality (Spector 2013), has been at the forefront of discussion of the role of AI in digital storytelling (Murray 2017), and has been the inspiration for a number of projects integrating narrative with visual and audio generation (Swartout et al. 2006; Marks, Estevez, and Connor 2014). These prior approaches have focused on graphics and visualization over the mechanisms of scene generation, the processes with which interactive applications create spatial environments from user input criteria.

For example, a Holodeck-inspired scene generation system could be used in the following scenario:

Sarah has an idea about a game about a detective. She is thinking of the events in the game but having a hard time imagining the space, so she uses the AI Holodeck to see how the space would look. "Holodeck, give me a detective's office". The Holodeck renders an office using objects sourced from a database. There is a desk and a chair, a window behind the desk, and a shelf in the corner. She then thinks that a desk lamp could make it more mysterious at night. "Holodeck, put a lamp on the desk." The Holodeck adds a lamp on the desk and a notebook beside it." She thinks "Yes, there should be a notebook on the desk!" But she feels the metal desk looks really rough in the office. She says, "Holodeck, give me a wooden desk". The Holodeck renders the new wooden desk by adding "wooden" to its initial database search of desks and replacing the original desk.

To facilitate the creation of scenes – and to populate them – some semantically annotated datasets are currently available that categorize objects with relative positions and sizes (Forbes and Choi 2017; Chang et al. 2015b).

Scene generation applications are able to parse these datasets to create a "scene template", a constrained mapping of a scene's objects and their basic spatial relationships between them (Chang, Savva, and Manning 2014). Systems like these use Natural Language Processing pipelines, such as the techniques in the CoreNLP library (Manning et al. 2014), to add items specified in a user's input text to a scene.

However, the addition of elements that are both 1) unspecified by a user and 2) gathered from semantically annotated datasets in order to maintain relevance to the user-described scene is missing from current research. Therefore, this paper aims to address the following research question: *Can semantically annotated datasets be used to extract context-informed scene templates in a text-to-scene generation application, including items not specified in the input text?*

In this paper, we introduce an AI Holodeck application. This system draws from previous NLP and scene generation work in order to create scenes with appropriate elements that were not specified by the user. For example, if the user inputs a *farm*, then the application may populate the scene with things commonly found in a farm such as cows, hay barrels, and fields of crops. The AI Holodeck can be used in a variety of scenarios: examples include visual story generation, game design and prototyping, interior design sketching or idea generation, and creating virtual or fantasy worlds.

The remainder of this paper explains how our system draws from and synthesizes existing works in the domains of natural language processing and scene generation, details the design of our AI Holodeck application, and discusses potential applications for integrating scenes generated by this system with other interactive mediums.

2 Related Work

2.1 Scene Recognition

Scene graphs have been extensively used and explored in the contexts of scene understanding and semantic image captioning (Johnson et al. 2015). Representing objects in a scene as nodes and their relationships as edges of graphs makes it possible to both represent the content of scenes, generate new scenes or manipulate scenes by modifying the corresponding scene graph (Dhamo et al. 2020). Once the objects and their relationships are learned, it can be used to meaningfully place objects in the scene. Scene graphs can be based on 3D scenes, reconstructed 3D scenes (Wald et al. 2020), images or text-image combinations.

Depending on the data format, different approaches have been devised to extract the scene content. SceneGen focuses on novel representations of scene graphs which embed positional and orientation information of a set of objects present in a given room to achieve the most realistic placement (Keshavarzi et al. 2020).

Other literature has investigated learning the implicit positional relationship between objects using transformers and attention mechanisms. In SceneFormer (Wang, Yeshwanth, and Nießner 2020), authors represent 3D objects and their corresponding environment through a sequence of numbers representing object category, location of objects in the environment, object orientation and dimensions of the room.

A recent transformer based model CLIP connects text and images to understand the image content. This discriminative model can be used to predict image content at scale by encoding the image and caption in parallel (Radford et al. 2021). This model in combination with generative models can be used to generate scenes (Galatolo, Cimino, and Vaglini 2021) but the effectiveness of this depends on the level of precision and detail required for the generated scene.

We infer the object categories and their positional relationships directly from images for two reasons: 1) images are more accessible than 3D models and there are more datasets available. 2) images function as representations of real world situations and convey properties that may be manipulated in 3D scenes such as *messy* desks or *cluttered* rooms.

2.2 Text-to-Scene Generation

WordsEye (Coyne and Sproat 2001), as one of earlier works on the text-to-scene generation, relies on explicit descriptions following the template of objects and their position. Requiring specific inputs in the format of "the [object] is [distance] [position] the [object]" make for a rigid and unnatural user experience.

Notating 3D datasets with natural language descriptions is one approach to improve the unnatural experience compared to the prior work (Chang et al. 2015a). However, the text query is not the only contributing component to achieve a natural user experience. SceneSeer breaks down the problem of text-to-scene generation into scene parsing, scene inference, scene generation and scene interaction.

To avoid unnatural languages caused by strict input language requirements like with WordsEye, they also bring objects that are not mentioned but are relevant to the mentioned objects. They select inferred objects by searching the object hierarchy and bringing the explicit object's parent objects with the highest probability (Chang et al. 2017). In addition to this, our approach considers the environment and objects that have the higher probability of co-appear with the explicit objects in that specific context. Our focus is not to produce the most sophisticated interior layout possible, but rather to demonstrate a natural language-based scene generation system that creates appropriate scenes fitting the user's input contextually and thematically.

2.3 Scene Manipulation

We explored scene manipulation research from the perspectives of *content* and *interaction*. In terms of content, research on scene manipulation focuses on scene level manipulation or object level manipulation which targets different purposes such as object removal or image blending.

Recent research investigates scene manipulation through updating the existing scene graph (Dhamo et al. 2020) and regenerating the scene. Other approaches explore modifying images using the semantic label maps or boundary maps extracted from the image (Wang et al. 2018). However, due to challenges presented by distinguishing different objects of the same type (e.g. several different cars in the scene), our approach is limited to single instances of an object in a scene.

In terms of interaction, prior research explored different methods or modes of interaction allowing users to manipulate scenes. SceneSeer (Chang et al. 2017) enables users to manipulate the scene with textual commands like "replace the bowl with a red lamp". In Scribbling Speech (Yang 2018), a speech-to-image generation tool, users interact with the interface through sound and modify the scene in a step by step process using natural language. The placement of objects happens in the different depths of the scene. We use the scene graph modification approach in our refactoring process and users can modify the scene by adding new queries.

Unlike the above tools, which visually render a scene, our approach also hosts the Holodeck component models in an API-like format, allowing for integration into a variety of applications such as Unity.

3 System Design

As seen in Figure 1, our Holodeck scene generator contains a full pipeline to collect input text and form a visual representation of a scene. Our system generates a scene template for any input text, determining objects and their locations in order for them to be placed in a scene. Then, implicit connections between objects in our semantically annotated datasets are used to add additional nodes to the scene template, creating a more vibrant scene. Objects are then mapped to a graphical representation of a scene in sequence, while resolving any collisions between them. Finally, a lightweight interface was created to allow for easily understood demonstration.

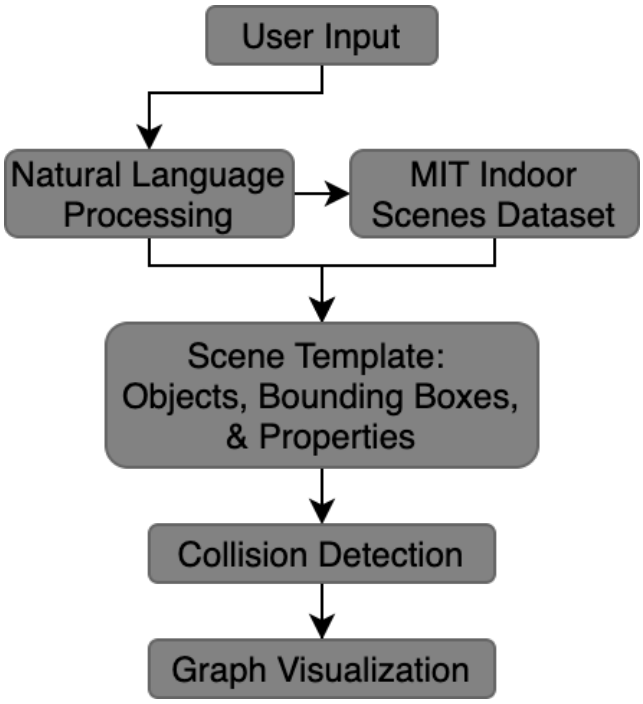

Figure 1: System flow for the AI Holodeck application.

3.1 Scene Templates

User input utterances are parsed through the CoreNLP library (Manning et al. 2014). The application separates sentences into dependency trees comprised of subjects, objects, and descriptors. Each subject and object are stored as a *node*, and each descriptor is stored as a *property* of that node. Descriptors concerning relative position between objects (such as "above" or "below") are stored inside of properties specifying a cardinal direction. Phrases such as "on top of" and "over" are all considered as the same "above" direction, and phrases as "beside" or "by" are set to either "left" or "right". These connections form a scene template (Chang, Savva, and Manning 2014), a collection of the various spatial relations between objects in a scene. The scene template allows all objects in a scene to be connected either directly or through an intermediate object, such as in Figure 2.

CoreNLP has made it possible to use an in-depth dependency parser, allowing for parsing complex sentence structures. However, we are also offering an offline version using the NLTK library (Bird, Klein, and Loper 2009) to extract objects, properties and their corresponding locations.

3.2 Implicit Nodes and Positional Relations

In order to create a more fleshed-out scene, our system adds additional nodes which are not explicitly mentioned by the user. Figure 3 provides an example of this concept. In this example, the user is creating an office space and has used the text "There is a couch and a table" as the first input. Since the "couch" and "table" are explicitly mentioned by the user, these nodes are created and used in the scene template. The addition of these *implicit* nodes allows the system

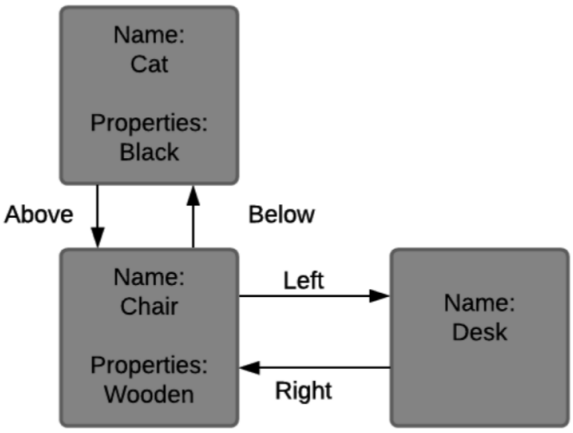

Figure 2: An example scene template, with inputs "There is a black cat on a wooden chair." and "The chair is to the left of the desk".

to bring in other objects such as a rug, a window, and a book because these are objects usually found near a couch or table in an office. If the object mentioned by the user is not usually found in such environment, eg. "a horse in an office", our system searches for other objects that have a high probability of being found in an office space rather than objects that are typically found near a horse.

With this method, the objects surrounding an object are dependent on which environment that object is in. Figures 3 and 4 show this difference by using the same input sentence "There is a couch and a table" in different environments of "bedroom" and "library".

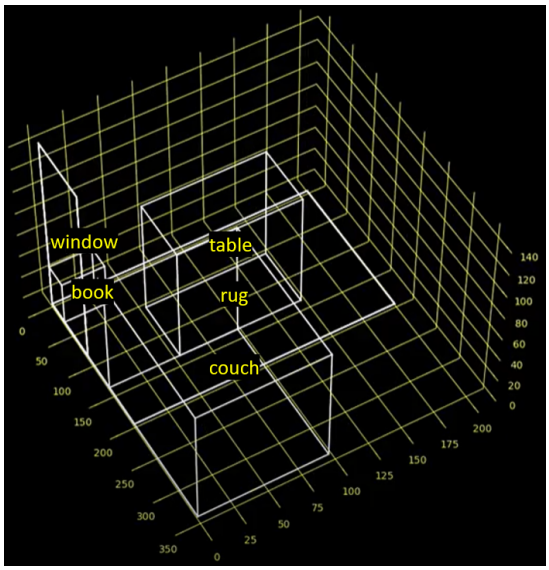

Figure 3: An example of adding implicit nodes, with the input sentence "There is a couch and a table." being used to create an office space.

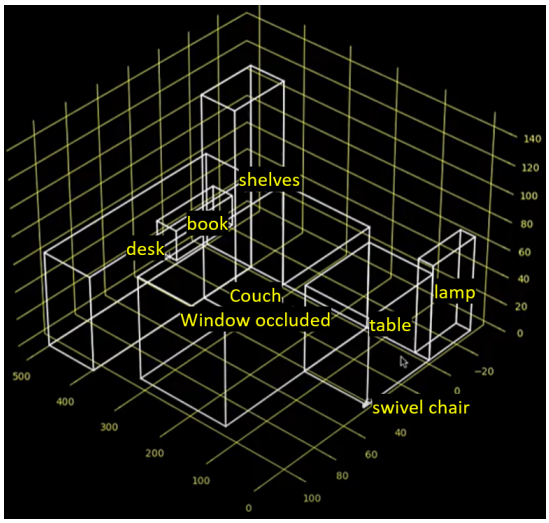

Figure 4: An example of adding implicit nodes. with the input sentence "There is a couch and a table." being used to create a library space.

Finally, our system prioritizes the explicitly defined positional relations over the implicit relations created by the system. Figure 5 shows that a computer is implicitly brought to the scene after 2 input sentences "There is a table" and "There is a chair". In Figure 6, the user adds the input "The computer is on top of the table", moving the computer to the explicitly specified position.

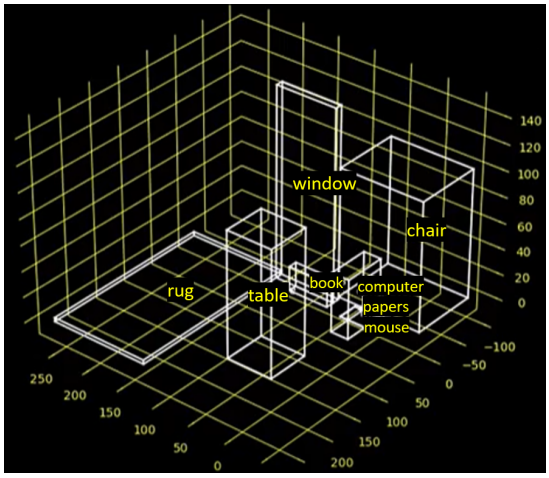

Figure 5: An example of adding implicit nodes. A computer is implicitly added to the scene.

Datasets Used for Extracting Implicit Relations In order to create a dataset of potential positional relations, we used the MIT Indoor Scenes Dataset (Quattoni and Torralba 2009), which contains 67 indoor categories and a total of 15620 annotated JPEG images. We sorted the objects found in each indoor category based on the number of occurrences in that category. Additionally, for each object found in a specific category, we looked at the objects found in immediate and far distance of the specified object. We divided these surrounding objects based on their positional relation to the specified object (eg. below, on top of) and sorted them based on the number of occurrences. We exported this information as a JSON file for our system's use.

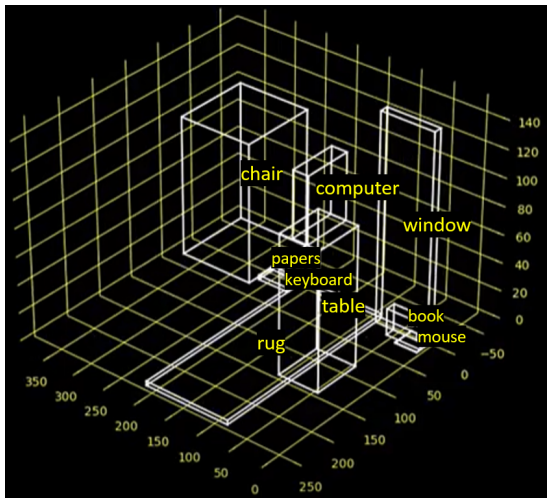

Figure 6: An example of prioritizing explicit relations to implicit ones. The user requests the computer to be moved to the top of the table.

3.3 Scene Visualization

When generating a scene, the system places bounding boxes representing each object in the scene template into a 3D graph. It searches for sizes for each object in the ShapeNetSem metadata. If none are found, they are replaced with default values for the output graph. The algorithm used to prioritize object placement queues objects on the *bottom* of a scene (objects with no "below" parameter), and recursively adds objects in those objects' "above" property on the graph, stacking objects on top of each other.

As each object is added to the graph, collisions are detected. Objects with lower priority (determined by their place in the scene template) are shifted in the direction corresponding to their property name the until their bounding boxes no longer overlap with the other object. For example, if one object is "above" the other, it will be shifted vertically.

3.4 Interface Design

The AI Holodeck application uses a Tkinter interface [1] (see Figure 7), activated from the command line.

The application opens a window with a menu for selecting a scene found in the Indoor Scenes dataset, a prompt for entering in text or microphone input, and a display of the objects currently registered from the scene. When a user selects the "Create Graph" button, objects found in the input text are added to the list of objects. The scene is then displayed as a movable, 3D matplotlib [2] graph in a separate window.

[1] https://docs.python.org/3/library/tkinter.html

[2] https://matplotlib.org/

The application allows for a number of command line arguments. *Mode* selects either text or voice input. When vocal input is activated, a recording button is added to the interface beside the text box. Pressing this button activates a continuous microphone stream until a sentence is recognized, which then populates the text field and automatically activates the graph creation function. *Model* selects either NLTK (Loper and Bird 2002) or CoreNLP (Manning et al. 2014) as a model to generate a dependency. The NLTK model is usable offline, while CoreNLP requires a separate command line prompt to begin a server with an internet connection. However, the CoreNLP model allows for more variety in sentence structure. Examination of future iterations of this system will include a comparison of error between the two models.

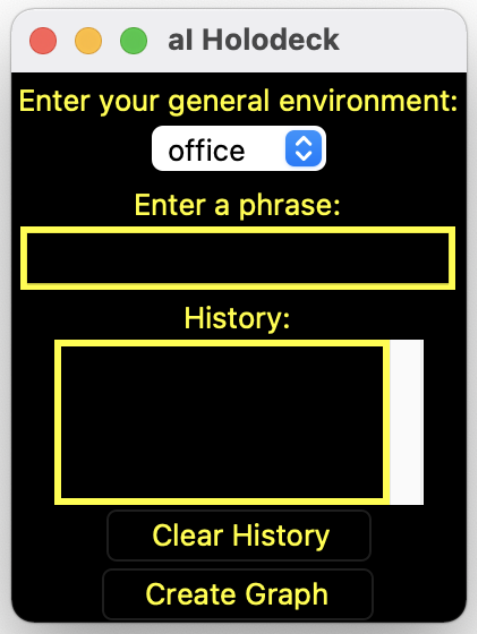

Figure 7: User interface for the AI Holodeck application

4 Discussion

A fully realized AI Holodeck application will require a relative positioning and collision detection system that allows for more spatial relationships than just "above", "below", "left", and "right". In particular, size-dependent relationships such as "inside" will allow generated scenes to have a greater amount of realism and variety.

This system is also limited in the fidelity of the visualizations it is able to create. Objects are represented only as a bounding box labeled with the object's name. A more sophisticated visualization application would include indexing of a database of 3D models, in order to dynamically populate generated scenes with appropriate representations of the objects inside.

Additionally, we plan on modifying the system to allow for the manual removal and repositioning of objects. As users correct the system output to fit the needs of the scene they are trying to create, the stored database will update with new spatial relationships and as such the system will be able to learn from generated scenes.

Our system is robust in terms of affording future modifications. For example, in this phase of the project, we have used the MIT dataset to extract possible positional relationships between various objects. These positional relationships can be constantly modified/added to by using other datasets not limited to the visual datasets, such as datasets of narrative texts. Other techniques – such as deep learning – could also be beneficial in removing our need for annotated visual datasets by extracting spatial relations automatically from other collections of images or narrative text.

The visualization of objects can be modified and extended to various platforms. Our system provides a scene template and graphical representation as formatted data and data structures extracted from the input text, which can be used by various platforms to create a detailed visualization. As explained in the system design of this paper, this data includes the various objects in the scene, their properties, positional relations and center-points for placement of the objects in the scene. Hence, the visual modification could be either in the form of changing visualization platform or in the form of adding new objects to the dataset of 2D/3D models used in these platforms.

5 Future Work

In future iterations of the system, we will include a separate narrative text interpretation module. This module will be comprised of a series of models trained on literature, which will provide additional scene details given a user's starting input. The current implementation, for both the NLTK and CoreNLP models, primarily uses simple sentences comprised of subject-object pairs in each clause. Training models on literature will enhance the system's ability to capture information from input sentences with a higher structural variety.

Other prospective improvements in the software are documentation and user functionality to facilitate connections to software such as Unity and virtual reality for integration with game development. This addition will be used to explore manipulation of objects already in a scene, and the placement or movement of objects with both input vocal commands and gestures. Additionally, once objects are generated in a higher-fidelity graphical environment, modifiers can be extracted from the input. These modifiers include adjectives describing the scene and the objects within. They will be added to the scene template and transmitted to the external environment in order to generate visuals more appropriate for the user query.

Future work will also include evaluation of the application. The first evaluation will include separate analysis of the NLP and scene generation models, in terms of precision in the sentences they parse as well as the relevance of objects generated for a scene. System performance will also be measured in terms of stability and framerate, while generating large amounts of objects. Finally, we will conduct user studies measuring the strength of the system as a piece of interactive technology as well as users' experience.

References

Bird, S.; Klein, E.; and Loper, E. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Chang, A.; Monroe, W.; Savva, M.; Potts, C.; and Manning, C. D. 2015a. Text to 3d scene generation with rich lexical grounding. *arXiv preprint arXiv:1505.06289* .

Chang, A.; Savva, M.; and Manning, C. D. 2014. Semantic parsing for text to 3d scene generation. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, 17–21.

Chang, A. X.; Eric, M.; Savva, M.; and Manning, C. D. 2017. SceneSeer: 3D scene design with natural language. *arXiv preprint arXiv:1703.00050* .

Chang, A. X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; et al. 2015b. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012* .

Coyne, B.; and Sproat, R. 2001. WordsEye: An automatic text-to-scene conversion system. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 487–496.

Dhamo, H.; Farshad, A.; Laina, I.; Navab, N.; Hager, G. D.; Tombari, F.; and Rupprecht, C. 2020. Semantic image manipulation using scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5213–5222.

Forbes, M.; and Choi, Y. 2017. Verb physics: Relative physical knowledge of actions and objects. *arXiv preprint arXiv:1706.03799* .

Galatolo, F. A.; Cimino, M. G.; and Vaglini, G. 2021. Generating images from caption and vice versa via CLIP-Guided Generative Latent Space Search. *arXiv preprint arXiv:2102.01645* .

Johnson, J.; Krishna, R.; Stark, M.; Li, L.-J.; Shamma, D. A.; Bernstein, M. S.; and Fei-Fei, L. 2015. Image retrieval using scene graphs. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3668–3678. doi:10.1109/CVPR.2015.7298990.

Keshavarzi, M.; Parikh, A.; Zhai, X.; Mao, M.; Caldas, L.; and Yang, A. 2020. Scenegen: Generative contextual scene augmentation using scene graph priors. *arXiv preprint arXiv:2009.12395* .

Loper, E.; and Bird, S. 2002. Nltk: The natural language toolkit. *arXiv preprint cs/0205028* .

Manning, C. D.; Surdeanu, M.; Bauer, J.; Finkel, J. R.; Bethard, S.; and McClosky, D. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 55–60.

Marks, S.; Estevez, J. E.; and Connor, A. M. 2014. Towards the Holodeck: fully immersive virtual reality visualisation of scientific and engineering data. In *Proceedings of the 29th International Conference on Image and Vision Computing New Zealand*, 42–47.

Murray, J. H. 2017. *Hamlet on the holodeck: The future of narrative in cyberspace*. MIT press.

Quattoni, A.; and Torralba, A. 2009. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 413–420. IEEE.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020* .

Spector, W. 2013. Holodeck: Holy Grail or Hollow Promise? Part 1. URL https://www.gamesindustry.biz/articles/2013-07-31-holodeck-holy-grail-or-hollow-promise-part-1.

Swartout, W.; Hill, R.; Gratch, J.; Johnson, W. L.; Kyriakakis, C.; LaBore, C.; Lindheim, R.; Marsella, S.; Miraglia, D.; and Moore, B. 2006. Toward the holodeck: Integrating graphics, sound, character and story. Technical report, UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY CA INST FOR CREATIVE ....

Wald, J.; Dhamo, H.; Navab, N.; and Tombari, F. 2020. Learning 3d semantic scene graphs from 3d indoor reconstructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3961–3970.

Wang, T.-C.; Liu, M.-Y.; Zhu, J.-Y.; Tao, A.; Kautz, J.; and Catanzaro, B. 2018. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8798–8807.

Wang, X.; Yeshwanth, C.; and Nießner, M. 2020. Sceneformer: Indoor scene generation with transformers. *arXiv preprint arXiv:2012.09793* .

Yang, X. 2018. Scribbling Speech. URL https://experiments.withgoogle.com/scribbling-speech.