**Dynamic generation of motion-based 3D virtual scenes**

Prepared For

Dr. Xiaobu Yuan

University of Windsor

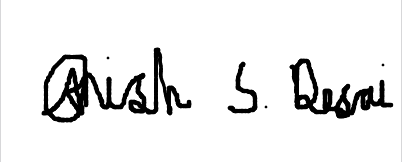
By

Anish Desai (104809484)

Kaitav Mehta (104911444)

10th April 2018

This survey is completely our own work, signed by

Anish Desai Kaitav Mehta

Contribution Table

|  |  |  |
| --- | --- | --- |
| Name | Role | Sign |
| Anish Desai  104809484 | Information gathering and literature survey.  Creating and modelling vehicles.  Adding interaction between objects.  Coding for generation and positioning of objects.  Documentation. |  |
| Kaitav Mehta  104911444 | Information gathering and literature survey.  Creating and modelling fire.  Adding sound to objects.  Coding for generation and after-effects of events.  Documentation. |  |

# Abstract

Virtual reality is quickly gaining importance in the field of education and healthcare. Virtual Reality Exposure Therapy (VRET) has been found to be effective in the treatment of patients suffering from Post-traumatic stress disorder (PTSD). This project aims to generate motion-based 3D scenes from text or speech. It uses the knowledge of spatial relation in addition to object and event mapping for generation of a scene from a parsed input text.

**Table of Contents**

[Abstract 3](#_Toc511127259)

[1. Introduction 6](#_Toc511127260)

[1.1 Post-Traumatic Stress Disorder (PTSD) 6](#_Toc511127261)

[1.2 Virtual Reality Exposure Therapy (VRET) 6](#_Toc511127262)

[2. Literature Survey 7](#_Toc511127263)

[2.1 Existing Applications 10](#_Toc511127264)

[3. System Introduction 11](#_Toc511127265)

[3.1 System Architecture 11](#_Toc511127266)

[3.2 System Component Interaction 13](#_Toc511127267)

[3.3 System Mechanisms 13](#_Toc511127268)

[3.3.1 Brute Force String Matching Algorithm 14](#_Toc511127269)

[3.3.2 Degeneralization 14](#_Toc511127270)

[3.3.3 Textualization 14](#_Toc511127271)

[3.3.4 Spatial Rule Based 14](#_Toc511127272)

[3.3.5 Event Mapping 15](#_Toc511127273)

[4. Software implementation 16](#_Toc511127274)

[4.1 User Input sub system 16](#_Toc511127275)

[4.2 Controller 17](#_Toc511127276)

[4.3 3D Objects 17](#_Toc511127277)

[4.4 Scripts 22](#_Toc511127278)

[4.5 3D Motion Generation 23](#_Toc511127279)

[4.6 Sound 23](#_Toc511127280)

[4.7 Virtual Screen 24](#_Toc511127281)

[5. User Manual 25](#_Toc511127282)

[5.1 Prerequisite 25](#_Toc511127283)

[5.2 Getting Started 25](#_Toc511127284)

[5.3 User Input 26](#_Toc511127285)

[5.4 Using newly created prefab 27](#_Toc511127286)

[6. Conclusion 29](#_Toc511127287)

[7. References 30](#_Toc511127288)

**List of Figures**

[Figure 1 - System Architecture Design 12](#_Toc511127054)

[Figure 2 - Flowchart showing the interaction between components 13](#_Toc511127055)

[Figure 3 - Textualization for ‘a cat is facing the wall’ [7]. 14](#_Toc511127056)

[Figure 4 - User Input Text Box 16](#_Toc511127057)

[Figure 5 - Properties of the Input Text Box 17](#_Toc511127058)

[Figure 6 - Various properties of a material 18](#_Toc511127059)

[Figure 7 - Texture properties in Unity 3D 19](#_Toc511127060)

[Figure 8 - Model properties in Unity 3D 20](#_Toc511127061)

[Figure 9 - Properties of a Prefab 21](#_Toc511127062)

[Figure 10 - Animation Component in Unity 23](#_Toc511127063)

[Figure 11 - Properties of an Audio Source 24](#_Toc511127064)

[Figure 12 - Initial Screen 25](#_Toc511127065)

# 1. Introduction

## 1.1 Post-Traumatic Stress Disorder (PTSD)

Post-traumatic stress disorder (PTSD) is a mental health condition that is triggered by a terrifying event — either experiencing it or witnessing it [1]. Symptoms may include flashbacks, nightmares and severe anxiety, as well as uncontrollable thoughts about the event. Most people who go through traumatic events may have temporary difficulty adjusting and coping. If PTSD is not cured, then its symptoms can interfere with day-to- day functioning of the patient [1]. People of all ages can have PTSD.

In addition to disturbing psychological symptoms, individuals that have experienced past trauma are also more likely to use alcohol or drugs. In fact, there seems to be a bidirectional relationship between substance use and PTSD [2]. PTSD is associated with poorer mental and physical health functioning, as well as increased levels of psychological distress.  These harmful consequences of the concurrent disorders can be quite pervasive and can negatively impact a variety of aspects of the individual’s life such as work, relationships, and outlook on life.  For these reasons, it is important that individuals dealing with PTSD get the help they need before their issues progress and become much worse [2].

There are several treatment options available to individuals experiencing addiction and co-occurring PTSD, including medication and psychological treatments. The average cost for First Year of Treatment with PTSD Combat veterans was $8300 in USA in 2011 [3]. In recent years Virtual Reality Exposure Therapy (VRET) has been one of the effective methods for treating PTSD patients. It has been shown by various research that VRET is efficacious for treating PTSD and helps in reducing depression, anxiety and anger [4] [5] [6].

## 1.2 Virtual Reality Exposure Therapy (VRET)

In VRET, doctors need to recreate the event in a virtual environment and analyze the health parameters of the patient when exposed to it. Thus, a software that helps in converting a text or speech into a realistic virtual scene can be highly beneficial for the doctors in treating people suffering from PTSD.

As per A. Rizzo and R. Shilling [4], by way of VRs capacity to immerse a user within an interactive computer-generated simulation, new possibilities exist that can go beyond the simple automation of previous clinical assessment and intervention approaches. Moreover, continuing advances in the underlying enabling technologies for creating and delivering VR applications have resulted in its recent widespread availability as a consumer product, sometimes at a very low cost. Thus, when one studies the scientific literature, examines the evolving state of the technology, and observes the growing enthusiasm for VR in the popular culture, the judgment for its future use in trauma, while appealing, deserves more research attention.

Although there exists software for treating war veterans suffering from PTSD through VRET, we aim to generalize this concept of VRET to treat all patients suffering from PTSD. Hence, this project aims at creating the virtual scenes with action from text-based input. With the help of this scene generation the doctors will be able to provide better and effective treatment.

# 2. Literature Survey

VRET involves generating realistic 3D virtual scenes from text or speech. The major elements of VRET include characters, virtual environments, audio, animation, and natural language processing. Here, we shall explore some of the previous works done in the generation of such virtual environments.

WordsEye [7] is a system for automatically converting text into representative 3D scenes. WordsEye relies on a large database of 3D models and poses to depict entities and actions. Every 3D model can have associated shape displacements, spatial tags, and functional properties to be used in the depiction process. WordsEye represents a new approach to creating 3D scenes and images.

StorVi [8] (story visualization) is a text-to-image conversion is a system that can visualize stories of multiple framing in pictures. The system focusses on fable stories for children ages 4-7 yrs. old. Recognizing the characters and partitioning of frames are the general problems of the study. In solving the two general problems, the researchers used classification algorithm/simple co-reference resolution algorithm and algorithm for recognizing the characters. They have also used the rule to partition the frames by sentence with character(s).

Text to 3D Scene Generation [9] introduces a dataset of 3D scenes annotated with natural language descriptions. The system learns from this data on how to ground textual descriptions to physical objects. The systems method successfully grounds a variety of lexical terms to concrete referents, improving 3D scene generation over previous works using purely rule-based methods. It evaluates the fidelity and plausibility of 3D scenes generated with the grounding approach through human judgments.

However, all the above systems lacked elements like sounds, moving images and in recognizing emotions of the characters in the story.

The CarSim System [10] is a prototype system to visualize and animate 3D scenes from car accident reports. It divides the problem of generating a 3D simulation into two subtasks: the linguistic analysis and the virtual scene generation. As a means of communication between these two modules, the researchers first designed a template to represent a written accident report. The CarSim system first processes written reports, gathers relevant information, and converts it into a formal description. Then, it creates the corresponding 3D scene and animates the vehicles. However, CarSim is limited is synthesizing only French and Swedish texts. Moreover, it is also limited in generating virtual environments of car accidents only.

Context Based Virtual Area Creation [11] has examples of systems and methods for context based virtual area creation. Some examples provide a quick and easy way for users to wrap virtual areas around contexts of interest. Examples of such contexts may be defined in terms of one or more of content, people, and real-world location. The virtual areas support real time communications between communicants (e.g., one or more of text chat, voice, video, application sharing, and file sharing) and provide a persistent historical repository for interactions in the virtual area.

SceneSeer [12] is an interactive text to 3D scene generation system that allows a user to design 3D scenes using natural language. A user provides input text from which the researchers extract explicit constraints on the objects that should appear in the scene. Given these explicit constraints, the system then uses a spatial knowledge base learned from an existing database of 3D scenes and 3D object models to infer an arrangement of the objects forming a natural scene matching the input description. Using textual commands, the user can then iteratively refine the created scene by adding, removing, replacing, and manipulating objects. It demonstrates how the generated scenes can be iteratively refined through simple natural language commands.

An interactive text to 3D scene generation [13] system learns the expected spatial layout of objects from data. A user provides input natural language text from which it extracts explicit constraints on the objects that should appear in the scene. Given these explicit constraints, the system then uses prior observations of spatial arrangements in a database of scenes to infer the most likely layout of the objects in the scene. Through further user interaction, the system gradually adjusts and improves its estimates of where objects should be placed. System then present example generated scenes and user interaction scenarios.

A similar system [14] addresses the grounding of natural language to concrete spatial constraints, and inference of implicit pragmatics in 3D environments. It presents a representation for common sense spatial knowledge and an approach to extract it from 3D scene data. In text-to-3D scene generation, a user provides as input natural language text from which we extract explicit constraints on the objects that should appear in the scene. The system mainly focusses on how to augment these explicit constraints with learned spatial knowledge to infer missing objects and likely layouts for the objects in the scene. It is demonstrated that spatial knowledge is useful for interpreting natural language and show examples of learned knowledge and generated 3D scenes.

The above two systems show the importance of spatial knowledge is the rendering of the objects and entities with relation to each other. Our project too emphasis on this spatial relation between the objects.

CONFUCIUS [15] was a research to investigate the process of mental imagery from a computational perspective, employing theories and resources from linguistics, natural language processing, and computer graphics about human language visualization. An intelligent storytelling system called CONFUCIUS, which visualizes single sentences into 3D animation, speech, and sound effects, has been implemented in Java and VRML. CONFUCIUS is an overall framework of language visualization, using computer graphics techniques with NLP to achieve high-level animation generation. CONFUCIUS gives promising results on word sense disambiguation (70% accuracy) about the dataset it is tested on.

BraveMind [4] presents the use of Virtual Reality as a clinical tool to address the assessment, prevention, and treatment of PTSD, based on the VR projects. It defines the clinical use of VR and describes a VR application designed for the delivery of prolonged exposure for treating Service Members and Veterans with combat- and sexual assault-related PTSD. It discusses details of early efforts to develop virtual human agent systems that serve the role of virtual patients for training the next generation of clinical providers, as healthcare guides that can be used to support anonymous access to trauma-relevant behavioral healthcare information, and as clinical interviewers capable of automated behavior analysis of users to infer psychological state. It concludes with a discussion of VR as a tool for breaking down barriers to care in addition to its direct application in assessment and intervention.

## 2.1 Existing Applications

|  |  |  |  |
| --- | --- | --- | --- |
| Application | Technology | Main operation | Is Motion a part of it |
| WordsEye [7], 2001 | SHRDLU system + PUT system | Manual mapping of objects + Spatial Relation | No |
| Text to 3D Scene Generation [9], 2015 | Stanford Parser | Automatically learns from data | No |
| StorVi [8], 2014 | WordsEye System | Similar to WordsEye, used for story telling | No |
| CarSim System [10], 2001 | VS Browser, VS Solver, VS Visualizer | Static Object, Dynamic Object, Collision | Yes |
| CONFUCIUS [15], 2006 | Java and VRML | Automatic generation through intelligence | Yes |
| BRAVEMIND [4], 2107 | Unity | Controlled VR war scenarios | Yes |
| Dynamic Scene Generation | Unity 3D | Manual mapping of objects, events + Spatial Relation | Yes |

# 3. System Introduction

The major goal of this system is to create realistic virtual scenes from text-based input. To make our system robust, scalable and efficient, we are using Unity 3D to develop virtual scenes. As per Radikal Labs [16], Unity 3D has many advantages as listed below.

Unity offers multi-platform usage benefit. Software developed in Unity 3D can be imported to iOS, PC, Web, Android, Mac and other game consoles. Unity is based on C# which makes it significantly competitive as the computing language is of extremely high level.

The Asset store is a power packed house of useful plugins and assets. Unity Physics is a brilliant part of the game engine where a built-in support system for PhysX physics engine is included. This helps in observing and developing real-time simulations based on thick ray casts, collision layers and skinned meshes.

Rendering in Unity engine depends on the graphic engines that use OpenGl ES, OpenGL and Direct3D for mobile platforms such as Android and iOS. Graphic development support on bump mapping, reflection and parallax is also provided. Global illumination, Light mapping and built-in path finding meshes along with video and audio backend system like Theora Codec and PostgreSQl provide extended support.

## 3.1 System Architecture

To define the structure and behavior of different components of the system, we have defined a system architecture which formally describes our system in an organized way. It is a conceptual model representation that supports reasoning about the structures and behavior of our system.

Figure 1 shows the interaction of various components of the proposed system. The input is a parsed text whose semantic structure can be RDF, WCDL or JSON as used by the K-parser [17]. Natural Language Processing can also be used to parse input stories and tokenizing key words of the story. This parsed text will contain entities, their properties and a sequence of events. This would be given as an input to the system.

The input will first be processed by the controller to derive entities, verbs and relations. The entities will be passed to the Object component where an object is generated for each entity. The location of these objects will be determined by the relationship determined by the controller. These relationships will also be passed to the Object component. Properties like size and color of the object will be set to default and can be changed when the controller identifies new attributes for the entities. Thus, the Object component generates objects with proper texture, coloring, size and position.

**https://lh4.googleusercontent.com/C3qcN4drpT-7mcliTvR6o-2nN0541bVjZNJBKQrz4OK9dfGUYonfolcq-aXGRKnuMgCvKSZa95wntcqVXhC4hiLV2NmkkBmnv3M-R07B8JiBa0kPEKpliFNN68RlVkuyuAyOTF4J**

Figure 1 - System Architecture Design

Based on the sequence of event, the controller initializes Event component to generate motion. All the actions that an object can perform in a virtual environment are defined either in scripts or in animation clips that are a part of the object. The Event component matches the correct action based on the event given by the controller and invokes that function in the script. Scripts can also determine the course of the action with additional functionalities like OnCollision (determining the course of an event when a collision is detected), FrameUpdate (determining the course of an event whenever a frame updates), etc. Moreover, the Event component will also consider physical laws (like the effect of gravity) and their consequences.

Like animation clips, audio clips can also be added to an object during an event. These audio clips make the virtual environment more realistic. The audio clips can also be controlled by the scripts which determine when to play or stop an audio. Once a motion is generated along with animations, actions and audio, it is rendered on the virtual screen. Rendering is generally done by the virtual reality software. The virtual screen should also have user interaction components so that the user can interact and decide the future series of events and new entities.

## 3.2 System Component Interaction

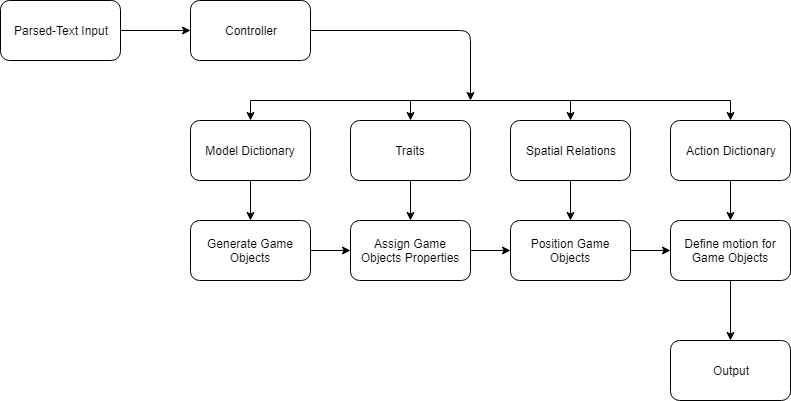


Figure 2 - Flowchart showing the interaction between components

Figure 2 shows how the input text will eventually be transformed into the output scene. The planning of visual elements and searching for the objects in the database will be done by the controller. If the object does not exist, then the process of degeneralization or textualization helps in displaying a realistic scene [8].

The spatial relations can be used to determine the position of the game objects [8] [7]. Apart from position, it can also contain information about the poses, grip and other attributes like size and color.

Although there are applications that generate static motionless scenes from text input [8] [7] [9], adding motion to the game objects is often seen as a challenging part. We are planning to add motion to the game objects using the technique of event mapping. As the input text will contain all the events in a sequential order, it can be mapped to various functions that can be invoked to define motion for game objects.

## 3.3 System Mechanisms

WordsEye [7] has mentioned various useful mechanisms to generate objects from parsed text. These mechanisms are discussed below.

### 3.3.1 Brute Force String Matching Algorithm

Brute Force String Matching Algorithm is an algorithm that checks every single word from the array of entities to match against the predetermined objects. This algorithm is used by identifying the corresponding objects of the possible characters or environment of the story in the database.

### 3.3.2 Degeneralization

Degeneralization is the act of specifying something that is general. In this process, if the collected text is a general categorical term, the system will search for the specific object instance of the same class.

Example: *“A vehicle is moving fast”.* In this example, the collected word *vehicle* is a general category. Since it doesn’t have an equivalent image in the database, the game object of a car will be used because it is an instance of the collected word *vehicle.*

### 3.3.3 Textualization

Textualization is the act or process of textualizing, i.e. rendering as text. The result of textualizing is a written version of the word. It is used when the input word doesn’t have an equal object in the database. The system will generate an extruded text of that word.



Figure 3 - Textualization for ‘a cat is facing the wall’ [7].

### 3.3.4 Spatial Rule Based

Spatial Relation specifies how some object is in space in relation to some reference object. In this technique, we come up with the rule that can identify the positions of the objects and that is called Spatial Rule Based. It is used to know the default positions of the objects in each sentence of the story. This is obtained by identifying the prepositions stated in the story like under, on, beyond, etc.

### 3.3.5 Event Mapping

Event Mapping defined the motion for the game objects. In this technique, every word from the array of action words of the input is searched in an action dictionary that contain a mapping between the word and the action. The action is usually a function that is invoked to add motion to the game objects.

# 4. Software implementation

System starts with the patient story-telling. The user of the system will be a doctor of the patient who is suffering from the post-traumatic stress disorder. The goal of the system is to generate a 3D animation for the doctors to understand the medical condition of the patient, so that doctor can provide the best treatment.

## 4.1 User Input sub system

The patient will describe a terrifying event, which affected his/her mental health condition. The event, in form of a story, will then be parsed using natural language processing into concepts, entities, keywords, categories, sentiment, emotion, relations, and semantic roles. Then the partially parsed text will be tokenized, and a final parsed input will be created which will have Objects, Events and Sounds filtered from the description of the patient story.

The above task will be achieved by software called, IBM Watson: Explore the Natural Language Understanding service. The Watson Natural Language Understanding service allows you to add the ability to your application to let you perform natural language processing that can enable advanced text analysis capabilities. The service offers developers the ability to analyze text to extract meta-data from content such as concepts, entities, keywords, categories, sentiment, emotion, relations, and semantic roles, all using natural language understanding. You can develop your own custom annotation models using Watson Knowledge Studio that will help you identify industry and domain specific entities and relations in unstructured text.

Currently, however, the system accepts only parsed text as input. The user input is a text box that appears on the top left corner of the game window when the software runs. The text box is an in-built GUI component of the Unity 3D.

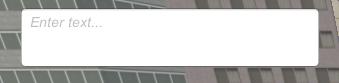


Figure 4 - User Input Text Box

Its dimensions and position are defined in the ‘Input Field’ object under the ‘Canvas’ object. The Inspector window displays its properties.

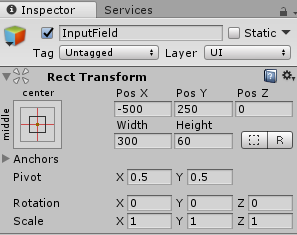


Figure 5 - Properties of the Input Text Box

## 4.2 Controller

The Controller is the most important subsystem. It gets the parsed input text from the user and then uses it create objects, their position and events on them. Controller coordinates with all other subsystem to ensure smooth functioning of the overall system. In our system, the controller is a C# script that interacts with other scripts and objects to generate meaningful scenes. There are functions defined in this script for generating objects and calling an action function.

## 4.3 3D Objects

Objects are the core elements of any virtual environment. Everything that is seen in a screen is an object. All the entities of the parsed input are displayed as objects in the virtual reality. All the events that arise from the parsed input are applied on the objects. Specific properties like color and size are also applied on the objects. An object defines the structural, physical and appearance properties of the entities. Thus, objects are essential subsystem of the virtual system.

In Unity 3D, Unity Assets store (https://assetstore.unity.com) is a power packed house of useful plugins and assets. These assets can be directly used as objects in the virtual environment. Objects can also be created through a combination of a wide range of models, textures, materials and prefabs.

* **Materials:** Materials are those components that defines the color, transparency, specular properties, metallic properties of the object. Unity 3D has a separate component where Materials and their properties can be easily defined. To create a new Material, use Assets->Create->Material from the main menu or the Project View context menu. By default, new materials are assigned the Standard Shader, with all map properties empty. Once the Material has been created, it can be applied to an object. Its properties can be changed in the Inspector. There are various shaders like FX (Lighting and glass effects), Nature (For trees and terrain), Toon (Cartoon-style rendering), and much more. The properties of a material are shown in Figure 4.

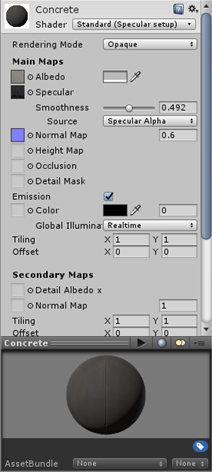


Figure 6 - Various properties of a material

* **Textures:** Textures are standard bitmap images that can be rendered on an object such that the appearance of the object resembles the appearance of the real-world entity. Textures can be reused for more than one object. Unity 3D has a separate folder in Assets folder to store all the images that are to be used as textures. Unity can import textures from most common image file formats. Properties of textures in Unity 3D are shown in Figure 5.

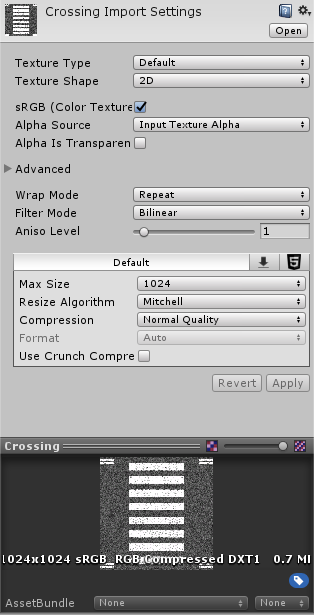


Figure 7 - Texture properties in Unity 3D

* **Models:**  Models are basic sub-parts of an object that have their own structure defined. Every Model defines a structure and other properties like Mesh Compression, Normal and Transform. Models can also use already defined materials to enhance its appearance. A model file can also contain animation data, which can be used to animate an object. The animation data is imported as one or more Animation Clips. Unity 3D supports a model component. Model files that are placed in the Assets folder in Unity project are automatically imported and stored as Unity Assets. A model file can contain a 3D model, such as a character, a building, or a piece of furniture. Properties of a Model in Unity 3D is shown in Figure 6.

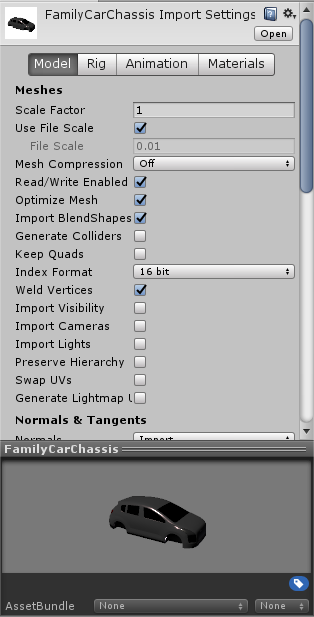


Figure 8 - Model properties in Unity 3D

* **Prefabs:** A Prefab is a reusable object. Prefabs can be inserted into any number of scenes, multiple times per scene. On adding a Prefab to a scene, an instance of it gets created. All Prefab instances are linked to the original Prefab and are essentially clones of it. No matter how many instances exists in a project, any changes made to the original Prefab will result in the changes being applied to all the instances. Properties of a Prefab are shown in Figure 7.

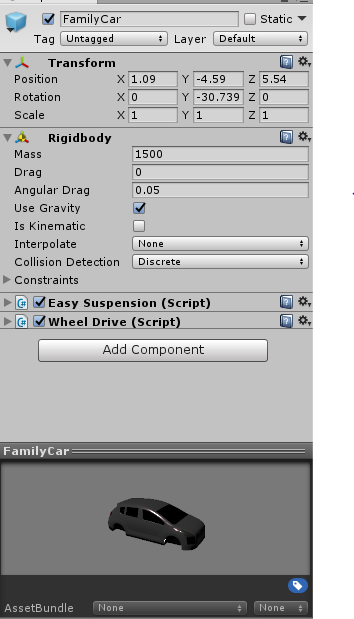


Figure 9 - Properties of a Prefab

* **Spatial Relations:** Spatial relations define the basic layout of scenes. They include relative positions, distances and orientations. Spatial relations are frequently an implicit part of actions and compound objects. They are often denoted by prepositions like on, under, beyond, etc. The exact placement of objects, in depictions of spatial relations, depends on the shapes and surfaces of those objects. Additionally, terms like in and under can have different possible spatial interpretations depending on the objects in question. Spatial tags for stems are applied to any object with a long, thin base leading to a thicker or wider top area.

## 4.4 Scripts

Scripting is an essential ingredient in all softwares. Even the simplest software needs scripts, to respond to input from the user and deduce the output from it. In Virtual environments, scripts can also be used to create graphical effects, control the physical behavior of objects or even implement a custom AI system for various objects in the game.

Unity 3D supports two programming languages natively:

* **C#** (pronounced C-sharp), an industry-standard language similar to Java or C++.
* **UnityScript**, a language designed specifically for use with Unity and modelled after JavaScript.

A script makes its connection with the internal workings of Unity by implementing a class which derives from the built-in class called MonoBehaviour. The Update function is the place to put code that will handle the frame update for the object. This might include movement, triggering actions and responding to user input, basically anything that needs to be handled over time during gameplay. To enable the Update function to do its work, it is often useful to be able to set up variables, read preferences and make connections with other objects before any action takes place. The Start function will be called by Unity before the Update function is called for the first time and is an ideal place to do any initialization. Unity also has various built-in functions that help in collision detection, coroutines, event system, etc.

In this project, the scripts can be found in the ‘Scripts’ folder, which is inside the ‘Assets’ folder. The Assets folder can be seen in the Project window. The scripts are .cs files as the scripting language used in this project is C#. Double click on the ‘GameController’ script file to open it in Visual Studio.

The GameController file represents the controller component of the software. It gets the input parsed string from the user and decodes it to identify entities and events. There are functions defined in this script for generating objects and calling an action function.

The ‘ModelDictionary’ script contains a mapping between key words of parsed entities and their corresponding objects. Thus, only those objects defined here will be generated during the runtime. It also has function defined for positioning game objects in relation to another game object.

Similarly, the ‘EventDictionary’ script contains a mapping of all the event that the controller will parse from the input text with their corresponding function generation motion. These motion-generating functions are defined in different scripts which represent the object in the virtual environment.

A script known as ‘SimpleCarController’ represents the car object. The ‘WheelDrive’ script represents the truck and SUV vehicles. Each of these scripts have functions defined in them to show how the vehicles will move, turn and at what speed. They contain the visual effects of the movement of the object. These scripts are also interlinked with their corresponding prefabs.

The ‘Entity’ script is a class that creates abstraction between the controller and the object scripts. Thus, the controller only invokes functions that generate objects or motion. The Entity class maps these events to their corresponding object.

## 4.5 3D Motion Generation

Motion is defined as any event that results in transformation to an object. This transformation can be in position, orientation, appearance, size, etc. A motion is generated for an object in the following three common ways: -

* **Animation:** Animation is a dynamic medium in which images or objects are manipulated to appear as moving images. It is achieved by a rapid succession of sequential images that minimally differ from each other. The physical movement of image parts through simple mechanics can also be considered animation. Unity’s Animation features include retargetable animations, full control of animation weights at runtime, event calling from within the animation playback, sophisticated state machine hierarchies and transitions, blend shapes for facial animations, and much more.

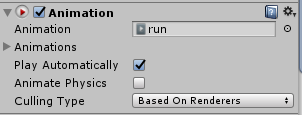


Figure 10 - Animation Component in Unity

* **Physics:** Physical forces can also be used to generate motion amongst various object in a virtual environment. E.g. Gravity can be used to make objects fall on a ground.
* **Scripts:** Scripting can be used to transform objects through code. One can add a force to move an object or displace an object by changing its position and rotation properties, or by destroying an object thereby making it disappear. Scripts can be used to showcase and reuse large number of motions effectively.

## 4.6 Sound

Audio is used to make the virtual environment more realistic. It greatly immerses a user into the virtual environment. Unity’s Audio features include full 3D spatial sound, real-time mixing and mastering, hierarchies of mixers, snapshots, predefined effects and much more. It can import most standard audio file formats (AIFF, WAV, MP3 and Ogg) and has sophisticated features for playing sounds in 3D space, optionally with effects like echo and filtering applied. Unity can also record audio from any available microphone on a user’s machine for use or for storage and transmission. Audio is attached to an object which becomes its Audio Source. Audio can be started, paused, resumed or stopped through scripts.

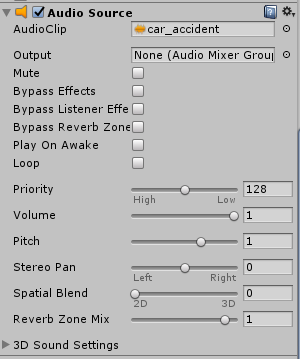


Figure 11 - Properties of an Audio Source

## 4.7 Virtual Screen

The final component of the virtual system is a screen that displays the virtual environment. A screen can be a monitor or a head mounted device - depending upon the level of immersion that is intended for the user. The virtual screen should also be interactive with the user. It should take in user input (through text or sound) and display its corresponding virtual environment changes. Unity has rich UI interface embodied in its Canvas object. The Canvas is the area such that all the UI elements should be inside it. The Canvas is itself an object with a Canvas component on it, and all UI elements must be children of such a Canvas. The UI elements of Canvas include text, image, button, toggle, slider, scrollbar, input field and dropdown. In our system we have shown an input box using Unity’s Canvas object.

# 5. User Manual

## 5.1 Prerequisite

* To use the technical project, you must have Unity 3D downloaded and installed in your PC/laptop. Download the latest version of Unity 3D from <https://store.unity.com>.
* Install the unity 3D software. For steps on installing the software, visit <https://docs.unity3d.com/Manual/InstallingUnity.html>.
* If you do not have Visual Studio, then the installer will install the Microsoft Visual Studio also. This is required for scripting in C#.
* Download the zip file and extract all its content to a location.

## 5.2 Getting Started

When you launch the Unity Editor, the Home Screen appears. If you have no existing Unity Projects on your computer, or if this is your first time opening the Unity Editor, the Home Screen displays the following Learn tab.

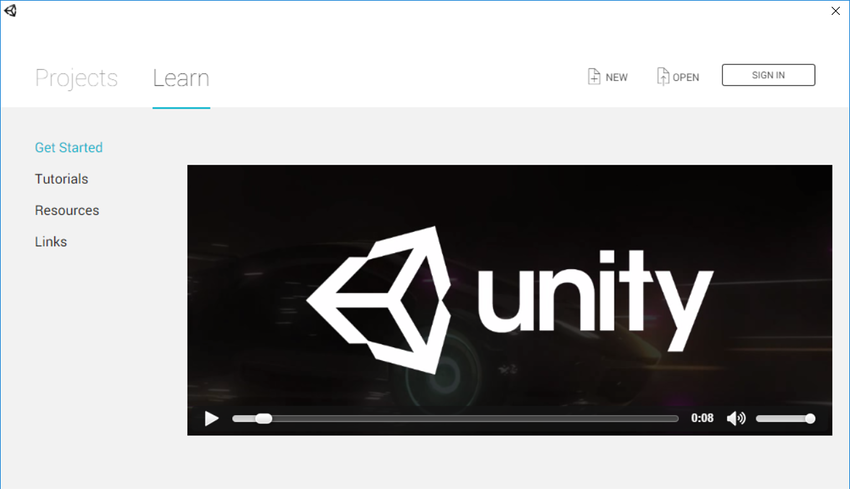


Figure - Initial Screen

Click on Open button on the top right side of the screen and open the folder that was extracted from the zip file. This would load the entire project into the unity software. This may take a few minutes.

Once the project is loaded, press the play button from the toolbar. The software is now running. In the input box, enter the following text

l, left,car,car

This would create two cars, one left of another, in the screen. Next enter the following text

l, right,truck

This would create a truck on the right side of the screen. Now enter

m, collide,2,3

This would make the truck and the car move towards each other, eventually colliding with each other. You would also see the car set on fire on collision. Also, an audio would be played when the collision takes place. Press the play button from the toolbar again to stop the software.

The above set of instructions was a quick demonstration of how the software works. The following chapters of this manual will explain how the system works and how to create your own objects and defining custom actions.

## 5.3 User Input

The user input is a text box that appears on the top left corner of the game window when the software runs. The text box is an in-built GUI component of the Unity 3D. Its dimensions and position are defined in the ‘Input Field’ object under the ‘Canvas’ object.

The user needs to enter the parsed text into the text box. The first letter of the parsed text should be either e, l or m. e resembles entity and will generate an object in the screen with a default position. Thus, the following text “e, car” will generate a car object at a default position.

l resembles location and is used to position an object in relation to another object. If the object is not yet created, it will first generate an object and then position it. Thus, “l, left,truck,car” will generate a car object and a truck object and it will position the truck on the left of the car object. The position of the car object will be the default position. Also, every object has an id which is the order in which it is created. Thus, “l, front,suv,2” will generate an SUV object in front of the second object. Giving “l, right,truck” as the input text will generate a truck object on the right side of the screen.

m resembles motion and is used to generate motion between various objects. Thus, giving “m, collide,1,2” as an input, the software will collide the first and second objects with each other. One must always use the id of the object in the parsed text when trying to generate motion as motion can be generated only for objects that are already created.

Thus, using the techniques of event and object mapping, one can generate various virtual scenes in this software. Also, more objects and events can be defined which can further enhance the scene generation and make the scenes more realistic.

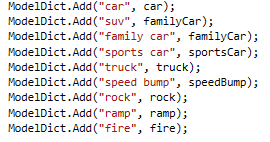
## 5.4 Using newly created prefab

Visit <https://docs.unity3d.com/Manual/Graphics.html> to gain more insight on how to model objects in Unity using  [Lighting](https://docs.unity3d.com/Manual/LightingOverview.html), [Cameras](https://docs.unity3d.com/Manual/CamerasOverview.html), [Materials, Shaders & Textures](https://docs.unity3d.com/Manual/Shaders.html), [Particles](https://docs.unity3d.com/Manual/ParticleSystems.html), [Visual Effects](https://docs.unity3d.com/Manual/comp-Effects.html), and much more. Once you have created a prefab (E.g. Bike), you are all set to use it in the project by following the below steps:

1. Make sure that the prefab is stored in the ‘Prefab’ folder inside the Assets folder.



1. Open the ‘ModelDictionary’ script. Add a new GameObject and giving a name (bike) to it. Also add a new mapping in the dictionary which maps the name with the GameObject. Save the file.



1. In the Project window, single click on the ModelDictionary file. The public variables of the file will be displayed in the Inspector window. The newly added GameObject will be shown there. Its value will be blank.



1. Drag and drop the newly created prefab (Bike) into the value field of the GameObject shown in the Inspector window.
2. Your new prefab is now ready to be used. Press the play button in the toolbar and enter the parsed text (e, bike) in the input box. Your newly created bike will now be displayed on the screen.

# 6. Conclusion

The purpose of this research was to help doctors to treat PTSD patients effectively, employing theories and resources from linguistics, natural language processing, computer graphics and animations about human language visualization. To conduct this research, an intelligent multimedia storytelling interpretation and presentation system was implemented. It creates 3D animations with speech from natural language texts. The benefits of the research include linking language modalities to visual modalities, automatic animation generation which combines procreated and dynamically generated (procedural) animation facilities into a unified mechanism, language ontology based on visual and auditory semantics, an overall framework of intelligent storytelling using computer graphics techniques with NLP to achieve high-level animation generation and a testbed for evaluating text-to-graphics applications. The evaluation results show that these approaches do contribute in automatic generation of virtual worlds from human natural language. Suggestions are made for future work to improve graphic quality, to extend the language knowledge base to cover more verb classes, to include discourse level analysis, to add human emotions analysis of the text, to view animation from different views and to port the system to other languages. Combined with a larger graphic library of 3D models and human animations the overall outcome of the work has the potential to impact on a wide variety of prospective areas such as virtual reality, computer games, movie production and direction, education, and intelligent multimedia.

# 7. References

|  |  |
| --- | --- |
| [1] | M. C. Staff, "Mayo Clinic," 25 October 2017. [Online]. Available: https://www.mayoclinic.org/diseases-conditions/post-traumatic-stress-disorder/symptoms-causes/syc-20355967. [Accessed 13 March 2018]. |
| [2] | I. Gavrysh, "Why It’s Important to Treat PTSD and Substance Abuse Together," *Moods Magazine,* p. 1, 1st June 2015. |
| [3] | D. W. Elmendorf, "The Veterans Health Administration’s Treatment of PTSD and Traumatic Brain Injury Among Recent Combat Veterans," The Congress of the United States, 2012. |
| [4] | A. Rizzo and R. Shilling, "Clinical Virtual Reality tools to advance the prevention, assessment, and treatment of PTSD," *European Journal of Psycotraumatology,* vol. 8, no. Supp 5 Article 1414560, 2017. |
| [5] | B. Rothbaum, L. Hodges, D. Ready, K. Graap and R. Alarcon, "Virtual Reality exposure therapy for Vietnam veterans with posttraumatic stress disorder," *The Journal of Clinical Psychiatry,* no. 62(8), pp. 617-622, 2001. |
| [6] | D. Beidel, B. Frueh, C. Bowers, S. Neer, B. Trachik, T. Uhde and A. Grubaugh, "Trauma management therapy with virtual-reality augmented exposure therapy for combat-related PTSD: A randomized controlled trial," *Journal of Anxiety Disorders,* 2017. |
| [7] | B. Coyne and R. Sproat, "WordsEye: An Automatic Text-to-Scene Conversion System," in *SIGGRAPH*, 2001. |
| [8] | K. Alcantara, J. Calandria, J. Calupas, J. Echas and R. Sagum, "StorVi (Story Visualization): A Text-to-Image Conversion," *International Journal of Future Computer and Communication,* vol. 3, no. 5, October 2014. |
| [9] | A. Chang, W. Monroe, M. Savva and C. Potts, "Text to 3D Scene Generation with Rich Lexical Grounding," arXiv preprint 1505.06289, 2015. |
| [10] | S. Dupuy, A. Egges, V. Legendre and P. Nugues, "Generating a 3D Simulation of a Car Accident from a Written Description in Natural Language: the CarSim System," in *TASIP*, 2001. |
| [11] | M. Josh, "Context Based Virtual Area Creation". United States of America Patent 9319357, 19 April 2016. |
| [12] | A. Chang, M. Eric, M. Savva and C. Manning, "SceneSeer: 3D Scene Design with Natural Language," 2017. |
| [13] | X. Chang, M. Savva and D. Manning, "Interactive learning of spatial knowledge for text to 3D scene generation," in *ACL 2014 Workshop on Interactive Language Learning, Visualization, and Interfaces (ACL-ILLVI)*, 2014. |
| [14] | X. Chang, M. Savva and D. Manning, "Learning spatial knowledge for text to 3D scene generation," in *EMNLP*, 2014. |
| [15] | M. Minhua, "Automatic Conversion of Natural Language to 3D Animation," 2006. |
| [16] | Radikal Labs, "Advantages of Game Development with Unity," 20 April 2016. [Online]. Available: http://www.radikal-labs.com/blog/advantages-of-game-development-with-unity/. [Accessed 11 February 2018]. |
| [17] | S. Aditya, Y. Yang, C. Baral, C. Fermuller and Y. Aloimonos, "From Images to Sentences through Scene Description Graphs using Commonsense Reasoning and Knowledge," arXiv preprint 1511.03292, 2015. |
| [18] | D. Plemenos and G. Miaoulis, "Theatrical Text to 3D Virtual Scenography," in *Intelligent Computer Graphics*, Springer e-book, 2012, pp. 23-40. |