

# **AUTOMATING LEARNER CENTRIC EDUCATION AND RESEARCH**

*Industrial Project submitted in fulfillment for the requirement of the Degree of*

## **MASTER OF TECHNOLOGY**

By

**KESHAN**



Department of Computer Science & Engineering

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

(Declared Deemed to be University U/S 3 of UGC Act)

A-10, SECTOR-62, NOIDA, INDIA

July 2023

@ Copyright JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

(Declared Deemed to be University U/S 3 of UGC Act)

NOIDA

July 2023

ALL RIGHTS RESERVED

Dedicated

to

My parents Dr. Manoj Srivastava and Prof. Rashmi Chandra

whose **COMPUTABLE VISION** is the foundation for my fastest **CEEVOLUTION**.

## TABLE OF CONTENTS

<b>CHAPTER-1.....</b>	<b>1</b>
<b>INTRODUCTION.....</b>	<b>1</b>
1.1 WHY AUTOMATION OF PERSONALIZED EDUCATION.....	1
1.2 WHAT TO BE AUTOMATED FOR PERSONALIZED EDUCATION .....	1
1.3 HOW TO AUTOMATE PERSONALIZED EDUCATION .....	2
<b>CHAPTER-2.....</b>	<b>3</b>
<b>LITERATURE REVIEW .....</b>	<b>3</b>
2.1 USER INPUT TO CODE .....	3
2.1.1 Text to Code Generation.....	3
2.1.2 Speech to Code Generation.....	5
2.1.3 Image to Code Generation .....	7
2.2 PERSONALIZED CONTENT CREATION.....	9
2.2.1 Speech to Speech Generation.....	9
2.2.2 Speech to Text Generation.....	11
2.2.3 Speech to Image Generation.....	12
2.2.4 Speech to Video Generation.....	13
2.2.5 Speech to 3D Generation.....	15
2.2.6 Text to Text Generation .....	17
2.2.7 Text to Speech Generation.....	19
2.2.8 Text to Image Generation .....	20
2.2.9 Text to Video Generation.....	22
2.2.10 Text to 3D Generation.....	23
2.2.11 Text or Speech to Animation .....	25
2.2.11.1 Text/Speech Driven Full-Body Animation .....	25
2.2.11.2 Text Based Compositional Animations.....	28
2.2.11.3 Voice Operated Character Animation (VOCA).....	29
2.3 PERSONALIZED CONTENT ORGANIZATION .....	30
2.4 INTERACTIVE LEARNING .....	32
2.4.1 BERT Style LLMs .....	32
2.4.2 GPT Style LLMs.....	34

2.4.3 Chat-REC .....	36
2.5 HAPTICS ENHANCED LEARNING.....	38
2.6 LIMITATIONS .....	39
<b>CHAPTER-3.....</b>	<b>40</b>
<b>NEW INITIATIVES.....</b>	<b>40</b>
3.1 CONTENT CREATION .....	40
3.2 STRUCTURE AND FUNCTIONING OF INTEGRATED VIRTUAL SCENE .....	45
3.2.1 Unity 3D Virtual Platform .....	45
3.2.2 Embedding the Teacher Bot in the Unity 3D Virtual Platform .....	52
3.3.3 Embedding the Video File in Unity 3D Virtual Platform .....	66
3.2.4 Hierarchy in the Integrated Virtual Scene.....	68
3.3 CONNECT WITH A FRIEND OR A TEACHER OR ANYONE .....	72
<b>CHAPTER-4.....</b>	<b>79</b>
<b>CONCLUSION AND FUTURE DIRECTIONS.....</b>	<b>79</b>
4.1 CONCLUSION.....	79
4.2 FUTURE DIRECTIONS .....	80

## LIST OF FIGURES

Figure 2.1: Timeline of LLMs.....	5
Figure 2.2: VCL Flowchart.....	6
Figure 2.3: Architecture of pix2code.....	9
Figure 2.4: AudioLM.....	10
Figure 2.5: Hierarchical Modeling of Acoustic and Semantic Tokens .....	10
Figure 2.6: Whisper AI.....	11
Figure 2.7: S2IGAN .....	13
Figure 2.8: Speech2Video.....	15
Figure 2.9: Referents Shown to Participants.....	17
Figure 2.10: Attention Mechanisms Used in LongT5.....	18
Figure 2.11: Emotional Text to Speech with BERT.....	20
Figure 2.12: unCLIP.....	22
Figure 2.13: Make-A-Video.....	23
Figure 2.14: Magic3D .....	25
Figure 2.15: Full-Body Animation Framework .....	27
Figure 2.16: Pose Discriminator with Hierarchical Two-Stream Model.....	29
Figure 2.17: VOCA Network Architecture .....	30
Figure 2.18: BERT’s Architecture .....	33
Figure 2.19: GPT-3 .....	35
Figure 2.20: Overview of Chat-REC.....	37
Figure 3.1: Default Blender Window .....	41
Figure 3.2: Editor Type Selector .....	42
Figure 3.3: Import Packages and Load Files.....	42
Figure 3.4: Material Colors Function .....	43
Figure 3.5: Cube Grid Function .....	43
Figure 3.6: Curve Object Function .....	44
Figure 3.7: Curves Between Points Function .....	44
Figure 3.8: Weights.....	44
Figure 3.9: Default Unity Window .....	45
Figure 3.10: Teacher Bot in Game View.....	53
Figure 3.11: State Transition Diagram of Idle Animation.....	55
Figure 3.12: State Transition Diagram of Interactive Animation.....	56
Figure 3.13: Idle Animation Properties .....	56
Figure 3.14: Head Nod Animation Properties.....	57
Figure 3.15: Talking Animation Properties.....	57
Figure 3.16: Idle -> Head Nod Transition .....	58
Figure 3.17: Head Nod -> Talking Transition.....	59
Figure 3.18: Talking -> Idle Transition .....	60
Figure 3.19: State Transition Diagram of Full Interactivity.....	61
Figure 3.20: Play Audio Script .....	61
Figure 3.21: Audio Selection .....	62

<b>Figure 3.22: Variable Declaration .....</b>	<b>62</b>
<b>Figure 3.23: Mic Initialization .....</b>	<b>63</b>
<b>Figure 3.24: Audio Clip .....</b>	<b>63</b>
<b>Figure 3.25: Loudness.....</b>	<b>64</b>
<b>Figure 3.26: Mic Start.....</b>	<b>64</b>
<b>Figure 3.27: Mic Stop .....</b>	<b>65</b>
<b>Figure 3.28: Start and Stop Mic when Application is at Focus .....</b>	<b>65</b>
<b>Figure 3.29: Video Player .....</b>	<b>66</b>
<b>Figure 3.30: Raw Image .....</b>	<b>67</b>
<b>Figure 3.31: Hierarchy Window .....</b>	<b>68</b>
<b>Figure 3.32: Hierarchy of Left Leg .....</b>	<b>69</b>
<b>Figure 3.33: Hierarchy of Right Leg .....</b>	<b>69</b>
<b>Figure 3.34: Hierarchy of Spine .....</b>	<b>70</b>
<b>Figure 3.35: Connect Button.....</b>	<b>73</b>
<b>Figure 3.36: Canvas .....</b>	<b>74</b>
<b>Figure 3.37: WebURL Script.....</b>	<b>74</b>
<b>Figure 3.38: Web Page HTML Script.....</b>	<b>75</b>
<b>Figure 3.39: Grids .....</b>	<b>76</b>
<b>Figure 3.40: Tiles.....</b>	<b>76</b>
<b>Figure 3.41: Classes .....</b>	<b>77</b>
<b>Figure 3.42: Web Page.....</b>	<b>77</b>
<b>Figure 3.43: Web App Creation .....</b>	<b>78</b>

## LIST OF TABLES

<b>Table 2.1: GPT vs BERT .....</b>	<b>32</b>
<b>Table 2.2: Fine Tuning .....</b>	<b>34</b>
<b>Table 3.1: Toolbar Control Tools .....</b>	<b>46</b>
<b>Table 3.2: Scene View Tools.....</b>	<b>48</b>
<b>Table 3.3: Overlays Tools.....</b>	<b>50</b>
<b>Table 3.4: Project Window Toolbar.....</b>	<b>52</b>
<b>Table 3.5: Animator Controls.....</b>	<b>54</b>
<b>Table 3.6: Transform Component of Main Camera .....</b>	<b>70</b>
<b>Table 3.7: Transform Component of Directional Light .....</b>	<b>71</b>
<b>Table 3.8: Transform Component of CNN, EventSystem and Teacher_Bot .....</b>	<b>71</b>
<b>Table 3.9: Transform Component of Hips .....</b>	<b>71</b>
<b>Table 3.10: Transform Components of LeftUpLeg .....</b>	<b>72</b>
<b>Table 3.11: Transform Components of RightUpLeg .....</b>	<b>72</b>
<b>Table 3.12: Transform Component of Spine.....</b>	<b>72</b>



## DECLARATION BY THE SCHOLAR

I hereby declare that the work reported in the M.Tech Industrial Project entitled “**Automating Learner Centric Education and Research**” submitted at **Jaypee Institute of Information Technology, Noida, India** is an authentic record of my work carried out under the supervision by **Prof Vikas Saxena, Head, Computer Science Department, JIIT, Noida; Prof Priya Ranjan Trivedi, President, Confederation of Indian Universities, New Delhi; and Prof Sandeep Marwah, President, Asian Academy of Film and Television, Noida**. I have not submitted this work elsewhere for any other degree or diploma. I am fully responsible for the contents of my M.Tech theses.

Name:

Department of Computer Science & Engineering and Information Technology  
Jaypee Institute of Information Technology, Noida, India

Date:

## SUPERVISOR'S CERTIFICATE

This is to certify that the work reported in the M.Tech Industrial Project entitled “**Automating Learner Centric Education and Research**” submitted by **KESHAN** at **Jaypee Institute of Information Technology, Noida, India**, is a bonafide record of his original work carried out under the supervision by Prof Vikas Saxena, Head, Computer Science Department, JIIT, Noida; Prof Priya Ranjan Trivedi, President, Confederation of Indian Universities, New Delhi; and Prof Sandeep Marwah, President, Asian Academy of Film and Television, Noida. This work has not been submitted elsewhere for any other degree or diploma.

Name:

Affiliation:

Date:

# INTERNSHIP CERTIFICATES

---

## CONFEDERATION OF INDIAN UNIVERSITIES

(Incorporated under the Central Act II of 1882 as a not-for-profit organisation)

A 42-43, Paryavaran Complex, South of Saket, New Delhi-110030

Email : ciu@ecology.edu + Website : www.universityindia.edu

Mobile/WhatsApp : 8527975833



Ref.: CIU/CERT/M.TECH/2023

20 July 2023

### INTERNSHIP CERTIFICATE

This is to certify that Keshan, M.Tech. student of CSE (AIML) at IIIT, Noida (Enrolment no 21303009), has done his 4<sup>th</sup> Semester Industrial Project on "Automating Learner Centric Education and Research" under the joint supervision by Prof. Vikas Saxena, Head, Computer Science Department, IIIT, Noida; Prof. Priya Ranjan Trivedi, President, Confederation of Indian Universities, New Delhi; and Prof. Sandeep Marwah, President, Asian Academy of Film and Television, Noida from 15<sup>th</sup> February 2023 to 20<sup>th</sup> July 2023.

We feel happy that Keshan focussed on creating innovative new solutions that will make big impact when the initiative matures in near future. His conduct and behaviour is very good and we wish him a grand success.

Prof. Priya Ranjan Trivedi  
President



## ASIAN ACADEMY OF FILM & TELEVISION

---

### MARWAH STUDIO COMPLEX

FC-14/15, Film City, Sector-16A, Noida-201301, U.P. India  
Tel. 0120-4831100, 2515254, 2515255, 2515256, Fax: 010-2515246  
E-Mail : info@aaft.com, Website : www.aaft.com

AAFT/Internship Certificate/2023-24/076

24<sup>th</sup> July 2023

### INTERNSHIP CERTIFICATE

This is to certify that Keshan, M.Tech student of CSE (AIML) at IIIT, Noida (Enrolment no 21303009), has done his 4th semester Industrial Project on "Automating Learner Centric Education and Research" under the joint supervision by Prof Vikas Saxena, Head, Computer Science Department, IIIT, Noida; Prof Priya Ranjan Trivedi, President, Confederation of Indian Universities, New Delhi; and Prof Sandeep Marwah, President, Asian Academy of Film and Television, Noida from 15th February 2023 to 20th July 2023.

I am happy that Keshan focused on new initiatives to integrate film medium with the personalized content in a virtual platform that will give new confidence to the least motivated minds also to learn and innovate just by watching the virtual videos interactively. I wish him all the best for his future endeavor.

Sandeep Marwah

President

## **ACKNOWLEDGEMENT**

I express my deep gratitude to my supervisors Prof Vikas Saxena, Prof Priya Ranjan Trivedi and Prof Sandeep Marwah for allowing me to work on this innovative project on "Automating Learner Centric Education and Research" conceived by my parents. All the supervisors had been very supportive and flexible to let me explore in my ways, listened to me carefully and encouraged me to do something meaningful.

I am also thankful to Tushar Gupta and Kritika Bhatnagar whose discussions with me helped to find practical solutions to move this project forward particularly when it was not clear what issues to be focused to complete within a semester.

I thank Jaypee Institute of Information Technology, Noida for providing me with the opportunity to work on this project.

## **ABSTRACT**

Emerging AI technologies are succeeding to let any person educate and innovate by recommending personalized content; curating content from a variety of sources; creating the desired content; rearranging the content at the web page for personalized experience; answering the questions for interactive learning; enhancing learning with the constructivist technologies, touch stimulating haptic wearables and quantum sensors; and converting user inputs to codes to let machine do any work etc. but there is no integrated virtual platform to let the user enjoy all such facilities just by giving instructions through speech or touch or thought. So I conceptualized and developed the idea of an integrated virtual scene that will facilitate to let anyone learn from her/his choice virtual human agent (e.g. learning special theory of relativity from virtual Einstein) and seek answers for her/his questions from the teacher bot in the voice of the teacher. This is done by creating an animated content in the blender software and converting it in the video format; embedding this video content in the Unity 3D virtual platform; and adding new functionalities to the animated character (teacher bot) to answer learner's questions in a sequential manner.

This initiative will shape a new education system that will give new confidence to the least motivated minds also to learn and innovate their best possible just by watching the virtual videos interactively. This will also create a new kind of UI different from that of a mobile or laptop, integrate artificial intelligence techniques in the virtual scene, facilitate mid-air 3d visualization as well as projection of the content on any screen or rough surface, and develop a very cheap wearable ring as an input-output device for quantum communication and control, automation and robotics, online manufacturing etc.; so the poorest person will also upgrade and enjoy the cloud and web based services.

# **CHAPTER-1**

## **INTRODUCTION**

Education is important to increase knowledge and understanding of the world, develop critical thinking and problem-solving skills, enhance creativity and innovation, improve social and emotional skills, and develop a stronger sense of civic engagement etc. So, it is necessary that everybody gets an opportunity to educate and upgrade her/his best possible.

### **1.1 WHY AUTOMATION OF PERSONALIZED EDUCATION**

- Institutionalized education will always upgrade few students only because of the limitations of infrastructure, materials and financial resources.
- Today's education is very costly while automated education can be personalized, global and free whose expenses can be borne from the online advertisements alone.
- Online experiments can fuel innovation by simulating and generating alternative ideas, models and technologies etc.

### **1.2 WHAT TO BE AUTOMATED FOR PERSONALIZED EDUCATION**

- AI can be used to recommend learning resources to anyone based on her/his cognitive capacity, goal and preference. It can curate content from a variety of sources as well as create content on demand.
- AI can present content in a virtual environment. It can rearrange the content (text, image, audio, video, animation etc.) at the web page according to user's preferences and can visualize it in the mid-air also.
- AI can be used to create interactive learning experiences. It can be done through personalized question answering; modeling and simulation; and constructivist learning like gamification, collaboration, discussion and role-playing etc.

- AI can enhance learning ability, attention and managerial capacity also by using vibrational devices, haptic wearables, and touch stimulating quantum sensors etc.

### **1.3 HOW TO AUTOMATE PERSONALIZED EDUCATION**

Historically, RNNs and LSTMs have been used for sequence transduction but they suffer from inability to parallelize due to sequential computation, short and long range dependencies, and linear distance between positions. In order to address some of these issues, researchers developed a method for paying ‘attention’ on particular words. The idea behind attention is that every word in a sentence might contain important information, so every word of the input must be taken into account for accurate decoding but the issue of parallelization is still not resolved with attention-based RNNs. So, CNNs were introduced for easy parallelization, exploitation of local dependencies, and logarithmic distance between positions but CNNs couldn’t manage the dependency issue with sentence translations. All this resulted in the development of Transformers to deal with the problem of Neural Machine Translation or Sequence Transduction.

Current LLMs like BERT and GPT use transformers in various ways to create personalized content and answer any learner’s question interactively. This is done by combining attention models with encoders and decoders to solve the issue of parallelization. GPT is a generative model that focuses on next word prediction while BERT is a discriminative model that focuses on masked word prediction. GPT uses constrained self-attention mechanism while BERT uses bidirectional self-attention mechanism to answer the user’s queries.



## CHAPTER-2

### LITERATURE REVIEW

Automation of education and research is rapidly evolving due to the emergence of new research ideas and technologies. In this chapter I briefly review the emerging technologies for

1. Converting user inputs to codes,
2. Creating personalized content,
3. Presenting personal web page in a virtual platform,
4. Answering learner's questions, and
5. Enhancing learning capacity using haptics.

## 2.1 USER INPUT TO CODE

### 2.1.1 Text to Code Generation

It is the process that automatically translates natural language descriptions into executable code. This can be done for a variety of programming languages. There are a number of different approaches for text to code generation, but most of them rely on Large Language Models (LLMs) to understand the natural language description and generate the corresponding code. LLMs are trained on massive datasets of text and code, which allows them to learn the relationships between natural language and programming concepts.

LLMs [1] are successful in code generation because of three reasons:

1. **Large model size:** Recent NL2Code LLMs have bigger sizes and better performance.
  - Regardless of size, the current models have room for development through further size increase.
  - Larger models typically have lower incidence of syntax errors.

- The production of semantically valid code is currently a constraint for large pre-trained models.
2. **Large and premium data:** The size of the training corpus grows along with the amount of the LLMs in the NL2Code field.
- This emphasizes how crucial it is to choose and prepare high-quality data.
  - Natural language and code data pairs that had been manually labelled were used to train early models. However, manual annotation requires a lot of work and takes time.
  - It is typical for LLMs to undertake data preparation on the sizable quantity of code in the acquired data to ensure the training corpus quality.
  - Pre-processing techniques aim to provide a code corpus which is clean, correct, complete, unduplicated, and broad in nature.
3. **Expert tuning:** Careful evaluation of numerous design decisions and hyper-parameters is necessary to train a great model.
- There is only a slight acceleration in convergence when initializing with different natural language models compared to starting from scratch.
  - In addition, a number of hyper-parameters, including sampling temperature, gradient accumulation steps, warm-up steps, window size, batch size, and learning rate require some expert tuning.
  - As the model gets bigger, the learning rate decreases.
  - Higher pass@100 and lower pass@1 are correlated with high temperature, indicating that LLMs produce more diversified predictions at higher temperatures and vice versa.
  - Sometimes a tiny model with a big window size performs better than a big model with a tiny window size.

- SentencePiece and Byte-Pair-Encoding are often the two strategies used by powerful LLMs to train a new tokenizer on code corpus. The separation of code content into tokens may be more accurate and efficient with a new tokenizer.

Figure 2.1 shows the timeline of different LLMs in chronological order.

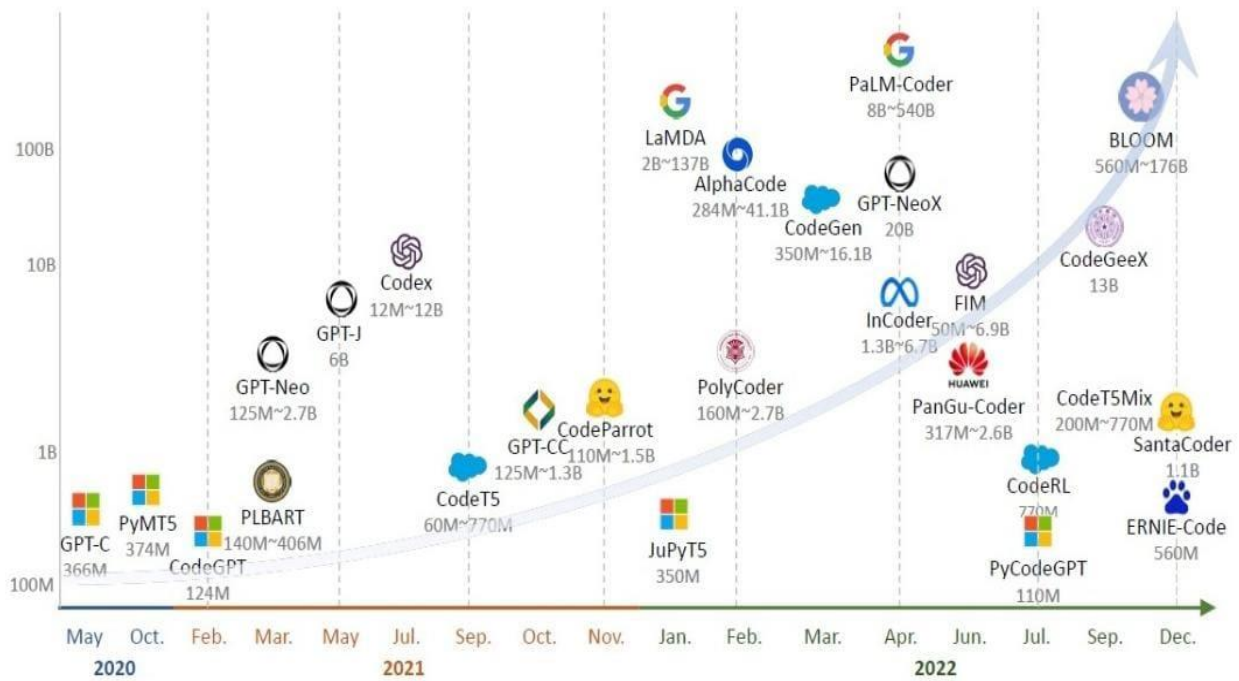


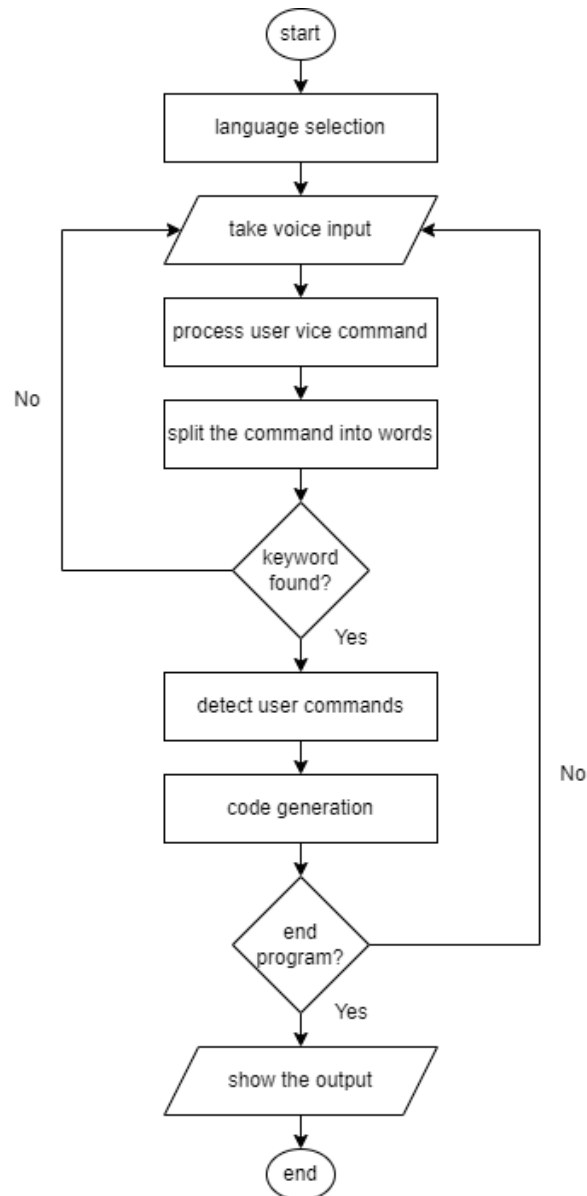
Figure 2.1: Timeline of LLMs [1]

### 2.1.2 Speech to Code Generation

It refers to the process of converting spoken language or speech into executable code. It is an emerging field that aims to enable anyone to write code by speaking rather than manually typing it. It utilizes Natural Language Processing (NLP) and programming language understanding techniques to interpret spoken instructions and generate corresponding code snippets.

One of the tools for speech to code generation is Voice Command Language (VCL) [2]. Figure 2.2 shows the working of VCL. The methodology of VCL is as follows:

1. A user communicates with the system to select the programming language of her/his choice using text to speech API of Google. The system also choses the Integrated Development Environment (IDE) where the coding operations will be performed.



**Figure 2.2:** VCL Flowchart [2]

2. The system captures user voice input using a microphone and saves it as an audio extension file with the help of speech recognition API of Google.

3. Using the Google's speech recognition API, the speech is converted into text format. In order to guarantee that the given command is correctly recognized, the text that is transformed is displayed in the user interface. This way the user is able to rectify the problem, if it arises.
4. The command is broken up into many words using NLP.
5. This system tries to locate a keyword that matches the command given by the user. The software's code section contains definitions for the keywords. If the matching keywords are found by the system then it will process to next steps otherwise it will go back to take new voice inputs from the user.
6. The system determines user requirements based on the matching keyword.
7. The system produces appropriate codes in response to the spoken command. Using the keyboard module of Python, the code is created in text format on the chosen IDE file.
8. The user has the option to either continue programming or execute the program.
9. Finally, the user issues a voice command to build and run the chosen IDE file.

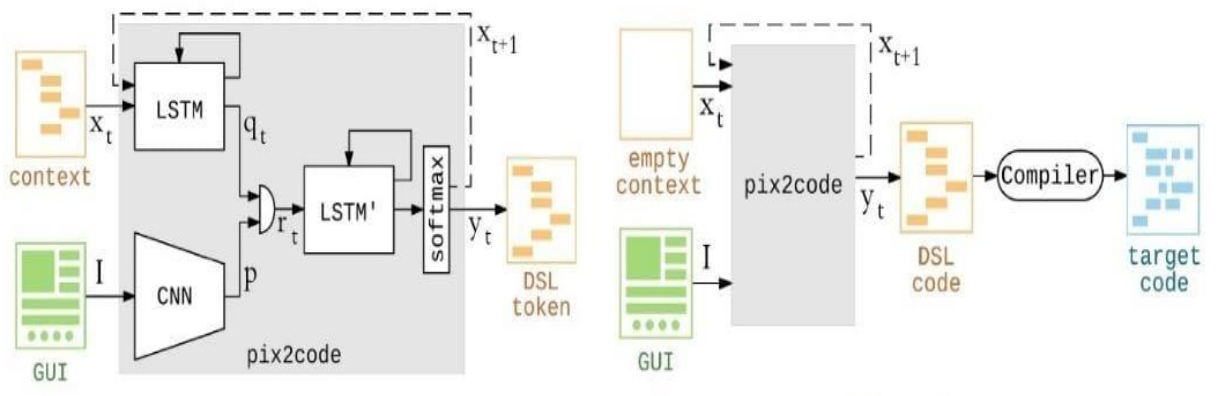
### 2.1.3 Image to Code Generation

It is the process of converting an image into executable code. It can be applied to automate the process of writing code or to produce more aesthetically pleasing or understandable code.

One of the tools for image to code generation is pix2code [3] that can be used to generate a code in user's choice programming language from a Graphical User Interface (GUI) screenshot just like a scene photograph can be used to generate textual descriptions in English. Figure 2.3 shows the architecture of pix2code. Its architecture is divided into three parts, namely:

1. **Vision model:** Understanding the scene that is being presented (GUI image) and determining the objects that are there, as well as their poses (such as element containers, labels, and buttons), positions, and identities.

- An input image is mapped to a vector (of fixed length) using CNN. The method used is unsupervised learning. As a result, this acts like an encoder.
  - Prior to being fed into the CNN, the input images are first reduced in size to  $256 \times 256$  pixels. There is no further pre-processing carried out.
  - Tiny  $3 \times 3$  receptive fields with stride 1 are convolved to get a fixed-size output vector by encoding each input image.
  - Before down-sampling with max-pooling, these processes are applied twice.
  - 1<sup>st</sup> convolutional layer has a width of 32, followed by layers with width of 64 and width of 128. The vision model ends with two  $1024 \times 1024$  completely connected layers that use Rectified Linear Unit (ReLU) as an activation function.
2. **Language model:** Generating semantically and syntactically correct illustrations by understanding the text (computer code).
- One-hot encoded vectors can be used, for a discrete input, to carry out token-level language modeling, doing away with the necessity for word embedding.
  - Traditional RNN architectures cannot simulate the interactions between data points dispersed throughout time periods due to exploding and vanishing gradients, so LSTM is used to rectify this problem.
3. **Decoder:** Utilize the answers to the two previous sub-problems to construct textual descriptions by using the latent variables deduced via scene understanding.
- The model can be optimized from end-to-end using gradient descent to forecast a token. This is done after seeing the image and preceding tokens.
  - Output's discrete nature reduces the work to a classification problem.
  - Candidate token's probability distribution is generated at each time step by the model's output layer. This enables multi-class classification with the help of a softmax layer.



**Figure 2.3:** Architecture of pix2code [3]

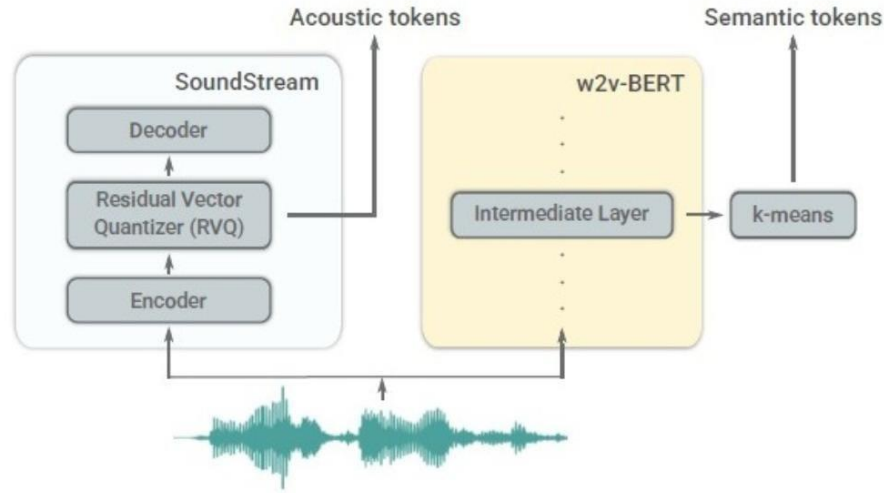
## 2.2 PERSONALIZED CONTENT CREATION

### 2.2.1 Speech to Speech Generation

It is a type of speech synthesis that converts one spoken utterance into another.

One of the tools for speech to speech generation is AudioLM [4]. A speech dataset is used for a comparison between acoustic and semantic tokens. This comparison shows that the reconstruction quality and phonetic discriminability of both the tokens is complementary. Without any textual explanations, the AudioLM produces speech that is coherent in terms of semantics, syntax and phonetics. Its model is as follows:

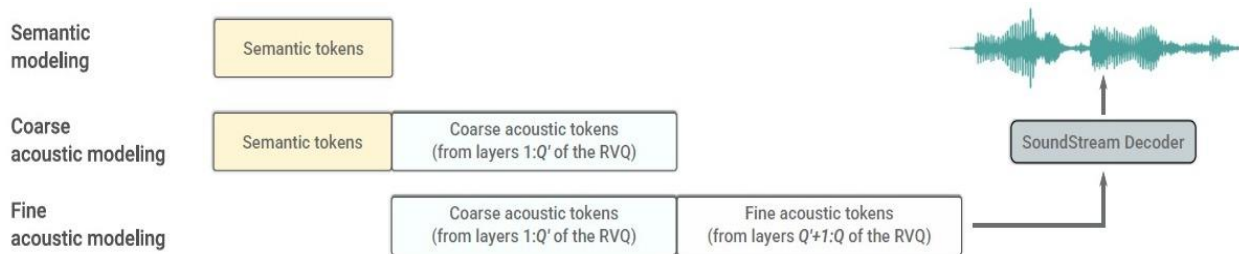
1. AudioLM framework has three components which process an audio sequence:
  - A tokenizer model maps  $y$  into a sequence  $x = \text{enc}(y)$  from a finite dictionary.
  - A transformer language model maximizes the likelihood by operating on the discrete tokens  $x$ . The model makes an autoregressive prediction of the token sequence at the time of inference.
  - A detokenizer model produces the waveform  $y = \text{dec}(x)$  by mapping the audio from the predicted tokens.



**Figure 2.4:** AudioLM [4]

2. Figure 2.4 shows the tokenizers used in AudioLM. Semantic and acoustic tokens provide long-term consistency and high-quality audio synthesis respectively when they are modelled inside the same framework. Figure 2.5 shows the three subsequent stages performed by AudioLM:

- **Semantic modeling:** It captures long-term temporal structure by modeling semantic tokens' autoregressive prediction.
- **Coarse acoustic modeling:** The hierarchical structure of the acoustic tokens is handled by flattening them in a row-major order.
- **Fine acoustic modeling:** This stage uses acoustic tokens that correspond to the fine quantizers, conditions them with the coarse tokens, and models the conditional probability distribution.



**Figure 2.5:** Hierarchical Modeling of Acoustic and Semantic Tokens [4]



## 2.2.2 Speech to Text Generation

It is the process of converting spoken language into text.

One of the tools for speech to text generation is Whisper [5]. Figure 2.6 show the approach used by Whisper. Its model is as follows:

1. All audio is re-sampled to 16,000 Hz and a Mel spectrogram representation is computed on 25 millisecond windows through a 10 milliseconds stride.

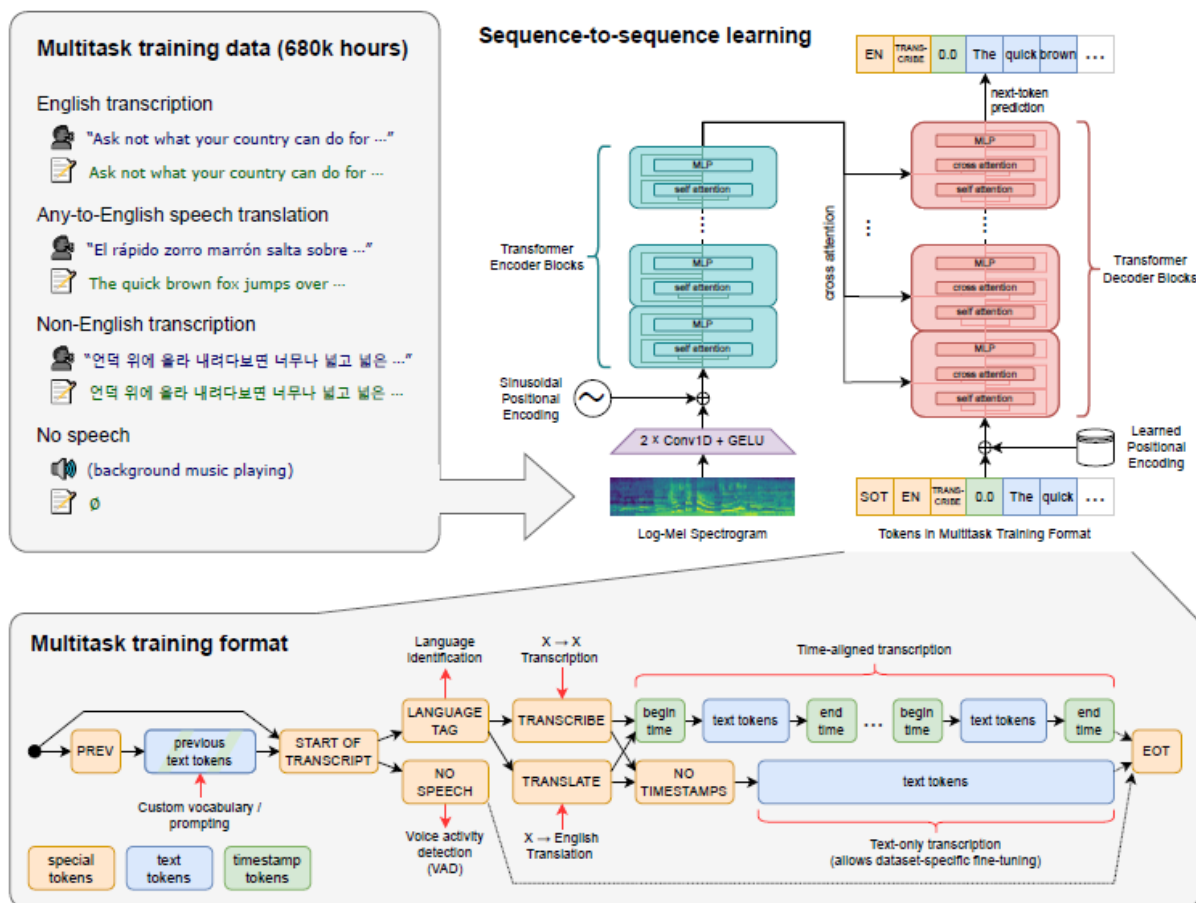


Figure 2.6: Whisper AI [5]

2. The input is scaled for feature normalization to lie between -1 and 1 with a mean of around zero beyond the dataset.

3. The encoder uses a tiny stem for input representation processing. This tiny stem consists of an activation function (GELU) and a filter (with width 3) in a two layer convolutional network where there is a stride of two in the second convolution layer.
4. The application of encoder transformer blocks results in stem output after sinusoidal position embeddings have been added.
5. The encoder output is subjected to a normalization of final layer after the pre-activation residual blocks are used by transformers.
6. The input-output token representations are tied together and position embeddings are learnt by the decoder.
7. The width and quantity of transformer blocks are the same for the encoder and decoder.

### 2.2.3 Speech to Image Generation

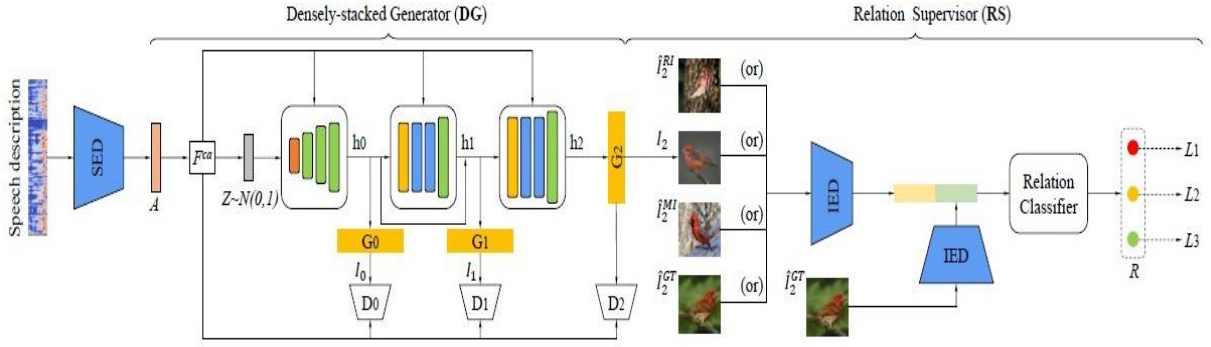
It refers to the process of image generation from spoken descriptions.

One of the tools for speech to image generation is S2IGAN [6]. Figure 2.7 shows the framework of S2IGAN. Its model is as follows:

1. Speech embeddings are created using Speech Embedding Network (SEN) and with the help of these speech embeddings, images are synthesized using Relation-supervised Densely-stacked Generative model (RDG).
2. The SEN is an encoder framework consisting of a Speech Encoder (SED) and an Image Encoder (IED).
  - Visual features are extracted through the IED. A single linear layer is then applied on top of it to transform the visual feature into a shared space of speech and visual embeddings. Consequently, an image embedding is obtained.
  - SED is made up of a self-attention layer, 2-layer bi-directional GRU, and a 2-layer 1-D convolution block. Mel spectrograms serve as an input of the SED.

3. The RDG consists of a Relation Supervisor (RS) and a Densely-stacked Generator (DG).

- The DG generates large scale images from small scale images through the hidden features. Non-linear transformations generate an image with the help of a generator.
- The RS ensures that the spoken descriptions are semantically aligned so that high-quality images are generated. An image set is formed having two classes. In one same class a real image, the ground-truth image and the generated fake image are present while in a randomly-sampled class a real image is present. Then, three relation classes are used to define the relation classifier.



**Figure 2.7:** S2IGAN [6]

## 2.2.4 Speech to Video Generation

It is the process of converting speech into video.

One of the tools for speech to video generation is Speech2Video [7]. Figure 2.8 shows the framework of Speech2Video. Its working is as follows:

1. Speech timbre, accent, and pace can all reveal valuable information about a speaker's appearance.
2. Disentangling the unique visual features from aural inputs is the key difficulty.
3. Speech is broken down into personal identity and emotional features.

4. Both the features are trained using distillation learning techniques from unlabeled talking videos.
5. Talking face frames are generated through an adversarial composer.
6. The adversarial composer and representation disentangler are trained separately.
7. Speech representation disentangling is done in the following manner:
  - The disentangling module is a distillation network that performs cross-modality.
  - An input is given in the form of unlabeled talking face video during the training stage.
  - Only the audio signals are used by the targeted representation extractor for learning.
  - Through a distillation network, the video's visual segments give the aural representation controlling data.
  - Different teacher networks are assigned, and the identification and emotional characteristics are simultaneously distinguished through the appropriate distillation process.
  - The teacher of emotion uses the Squeeze and Excitation Network (SENet) while the teacher of identity uses Visual Geometry Group Network (VGGNet).
  - A pre-trained module (contrastive prediction coding) specifically extracts from the speech an extensive representation of audio information.
  - The identity and emotional features are respectively distilled by the two student networks.
8. A talking face video is generated from encoded vectors. An adversarial decoder is used for the transformation.
  - Three feature vectors (identity feature, emotion feature and the frame feature sequence) have been extracted from the encoder.
  - The emotion feature is immediately concatenated for transposed convolution for each frame.

- The identification feature, on the other hand, is handled separately for improved feature supervision.
- Identity feature specifically passes through three convolution layers (that are transposed), with twice the height and width.
- A 1x1 convolution aligns the geometry of each intermediate feature map to the nearly matching layer in a VGGNet.
- The identity features processed independently significantly improved the ability to distinguish between different people in the final video frames.

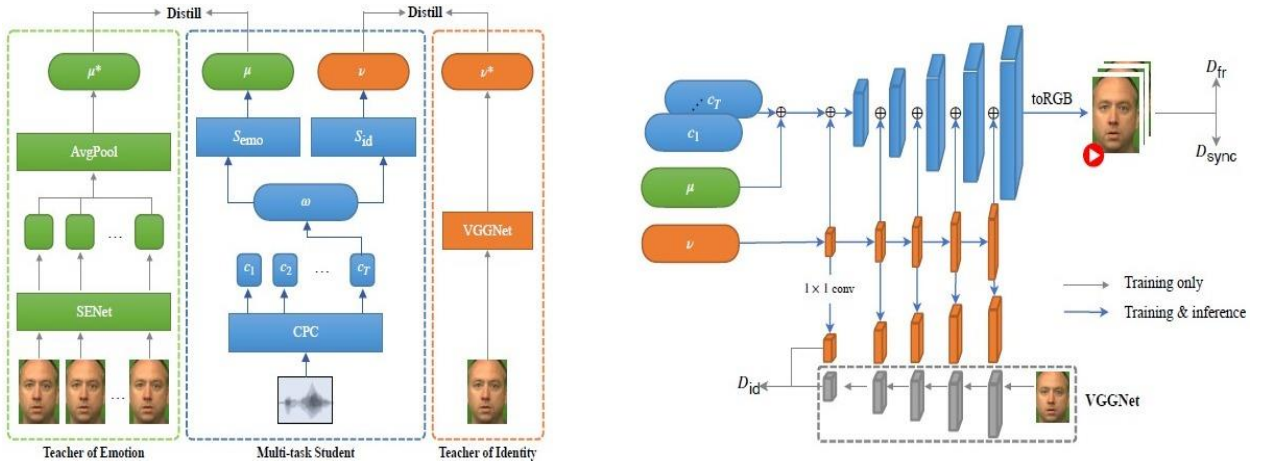


Figure 2.8: Speech2Video [7]

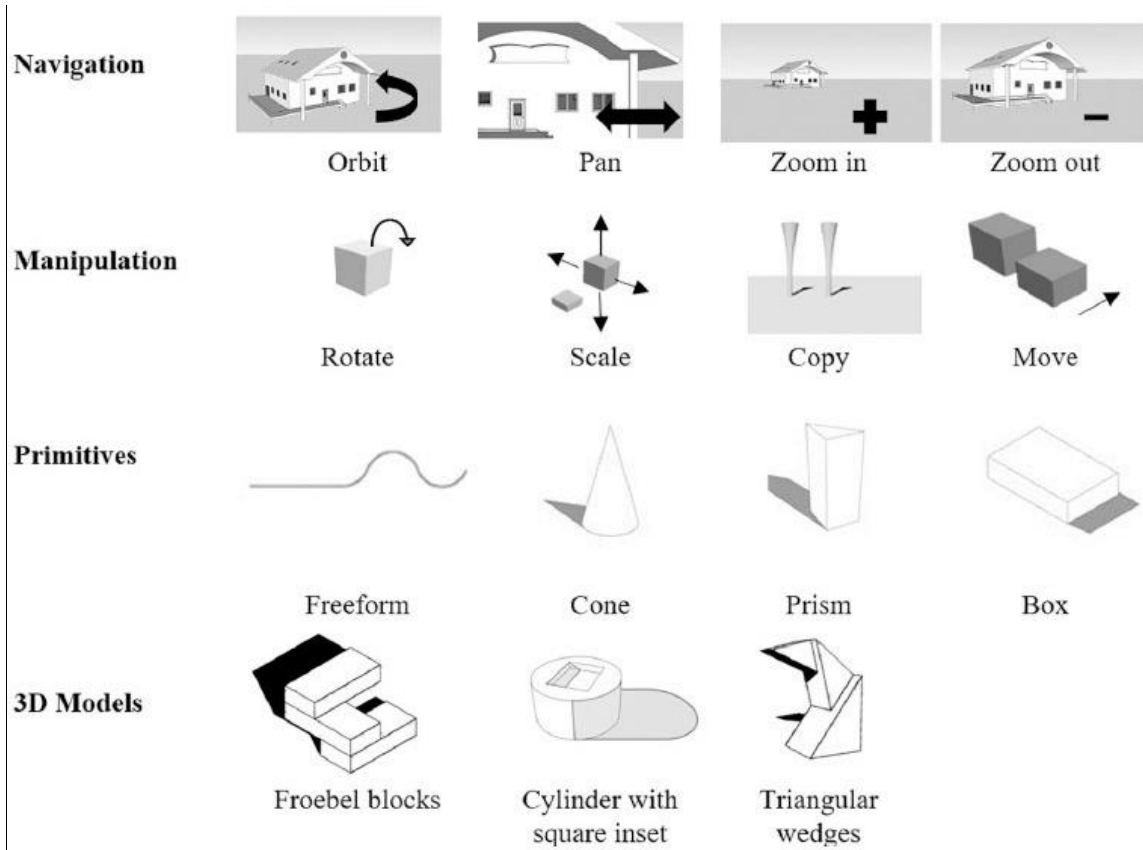
### 2.2.5 Speech to 3D Generation

It is the process of converting speech descriptions into 3D models of scenes and objects.

The experiment's [8] goal was to elicit gestures and verbal command phrases for 3D CAD modelling tasks like manipulating and seeing 3D objects. Each participant in the experiment received individual attention over the course of two 20-minute sessions. Participants watched brief 3D CAD video clips of modelling jobs or individual photos of 3D CAD items on a 50-inch screen that was three meters away from them. The referents were elementary CAD modelling assignments. Each referent was displayed for roughly 15 seconds. Referents are divided into three groups:

1. **Navigation:** The act of guiding the user's vision across an area is referred to as navigation in CAD. Participants were shown the following navigational reference objects: zoom out, zoom in, pan, and orbit.
  - A video clip was shown to participants for the referent orbit in which the view rotated around a house in anti-clockwise direction.
  - A video clip was shown to participants for the referent pan that included a house's front façade being panned across from right to left.
  - A video clip was shown to participants for the referents zoom out and zoom in, respectively in which a house can be seen from a distance before closing in on it, and vice-versa.
2. **Manipulation:** In CAD modelling, manipulation refers to procedures for changing an object's attributes, like its scale, orientation, or position. This group's categories include: move, copy, scale and rotate.
  - A video clip was shown to the participants in the rotate task that involved a box rotating 45 degrees to the right.
  - A video clip was shown to the participants in the scale task that had two boxes in 1<sup>st</sup> frame. The larger box doubled in size when the video clip started playing, scaling up equally.
  - The video clip for copy displayed a composite item that cloned towards the right.
  - Participants in the move experiment watched a video clip with two boxes in 1<sup>st</sup> frame. The smaller box shifted along the x-axis.
3. **Primitives:** This group's categories included box, prism, cone, and freeform. The participants were simply shown still images.

Figure 2.9 shows the various referents shown to participants.



**Figure 2.9:** Referents Shown to Participants [8]

## 2.2.6 Text to Text Generation

It is the process of automatically generating text from a given text prompt or input.

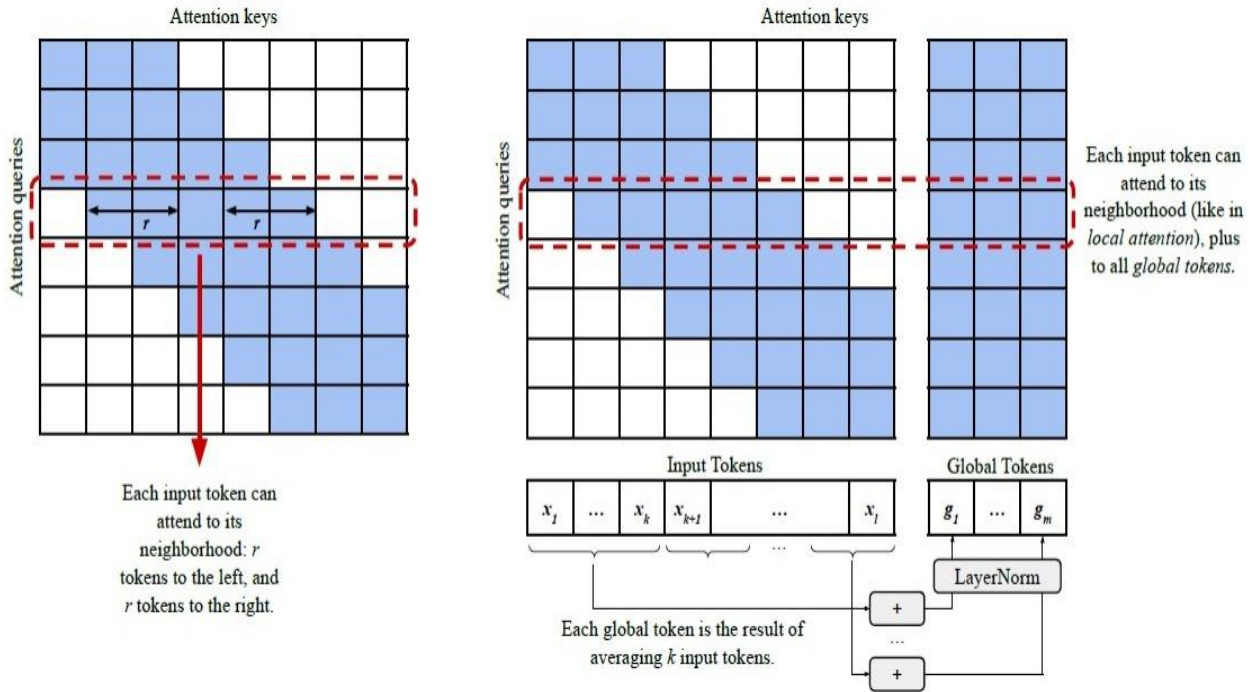
One of the tools for text to text generation is LongT5 [9]. The attention mechanism is the primary architectural distinction between original T5 and LongT5. Transient Global attention (TGlocal) and Local attention are the two attention mechanisms used in LongT5. Both variations maintain a number of T5 characteristics, including packing support and relative position representations as well as checkpoint compatibility of T5. Figure 2.10 shows the attention mechanisms used in LongT5. The architecture of LongT5 is as follows:

1. **Local attention:** Local attention operation, with a sparse sliding-window, substitutes the encoder self-attention action in T5.

- In particular, this formulation limits each token's ability to attend to  $r$  tokens to its left and right for a specified local radius  $r$ .
- It is discovered that  $r = 127$ , where  $r$  is the quantity of adjacent tokens to the right and left, was adequate in practice.
- Complexity scales linearly with the length of the input sequence for a given value of  $r$ .

2. **Transient global attention:** The input sequence is broken down into  $k$  tokens and a global token is computed for each block by adding up the embeddings of each token. There are only two additional parameters added by TGlobal attention:

- Relative position biases, in T5-style, reflect the distance between the block of an input token and the block of each global token it is attending to.
- Normalization parameters for each global token's embedding in a T5-style layer.



**Figure 2.10:** Attention Mechanisms Used in LongT5 [9]



### **2.2.7 Text to Speech Generation**

It refers to the process of converting written text into spoken audio.

Neural text to speech systems [10] create audio files from a text source using deep learning techniques. Essentially, it consists of a frontend and a backend. In the frontend, the text is normalized (written form to verbal form) using a text normalizer and this normalized text undergoes the grapheme (spelling) to phoneme (pronunciation) conversion process. In the backend, the phonemes are fed to an acoustic model. The model then converts the phonemes to Mel spectrogram. These spectrograms are then converted to their corresponding waveform through a vocoder (voice encoder). Figure 2.11 shows the model of Text aware Emotional Text to Speech with BERT.

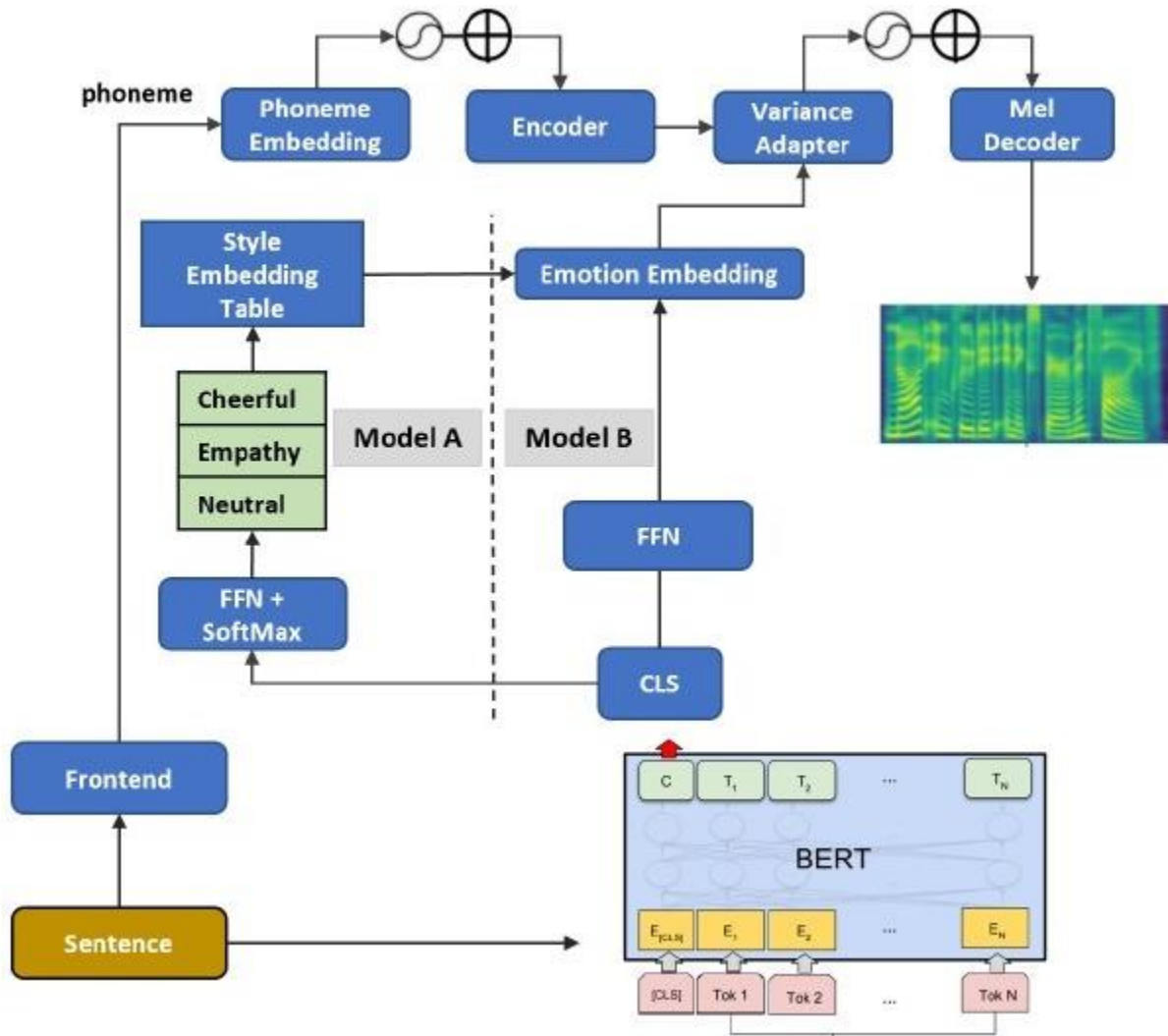
The structure of Text aware Emotional Text to Speech with BERT is as follows:

#### **1. Text emotion modeling through BERT:**

- The entire text is represented using the first token's [CLS] final hidden states. This representation is now referred to as a BERT embedding.
- An emotion label's probability is then predicted by layering a softmax activation function and a conventional feed-forward layer in addition to BERT embedding.
- Recommended sentiment mapping are used to map empathy, neutral, cheerful, negative to cheerful and positive emotions respectively.

#### **2. Emotional text to speech with BERT:**

- The syntactic and semantic meaning of a given text are both captured by [CLS] token embedding and bidirectional representation of text is encoded by BERT.
- BERT model fine-tuned on a classification task of textual emotion leads to a variety of embeddings that can capture the nuances of the text's emotions.



**Figure 2.11:** Emotional Text to Speech with BERT [10]

## 2.2.8 Text to Image Generation

It is the process of converting a text into an image.

One of the tools for text to image generation is unCLIP [11]. Figure 2.12 shows its overview.

A two-stage model with a prior and a decoder is used to generate images from text. A CLIP image embedding is generated through a prior when a text caption is already provided while an image is generated by a decoder with image embedding conditioning.

The decoders, trained on pictorial representations, are capable of creating different images while maintaining its semantics and style and changing the non-essential aspects that were left out of the representation. Additionally, CLIP's joint embedding space allows for zero-shot language-guided image transformations.

To invert the CLIP image encoder, a diffusion decoder is trained. A non-deterministic inverter can generate various images that correspond to a specific image embedding. The availability of an encoder and a decoder enables functionality beyond text to image conversion.

The training dataset make up the pairs  $(l, m)$  of images  $l$  and their matching captions  $m$ . Let  $n_t$  and  $n_i$  be the respective text embeddings and CLIP image of an image. A prior and a decoder are used to generate images from captions:

- A prior  $P(n_i|m)$  generates CLIP image embeddings based on captions. It uses diffusion as well as autoregressive model.
- A decoder  $P(l|n_i, m)$  creates images conditioned on CLIP image embeddings (and optional text captions). It only uses diffusion model.

Given the CLIP image embeddings of the photos, the decoder inverts the images, whilst the prior learns a generative model of the image embeddings themselves. These two elements combined together create a generative model  $P(l|m)$  of images with provided captions:

$$P(l|m) = P(l|n_i, m) = P(l|n_i, m) P(n_i|m)$$

CLIP image embeddings is a deterministic function of an image, so the 1<sup>st</sup> equality holds and chain rule is responsible for holding the 2<sup>nd</sup> equality.

So, using the prior to sample CLIP image embeddings and the decoder to sample images, we can sample from the real conditional distribution  $P(l|m)$ .

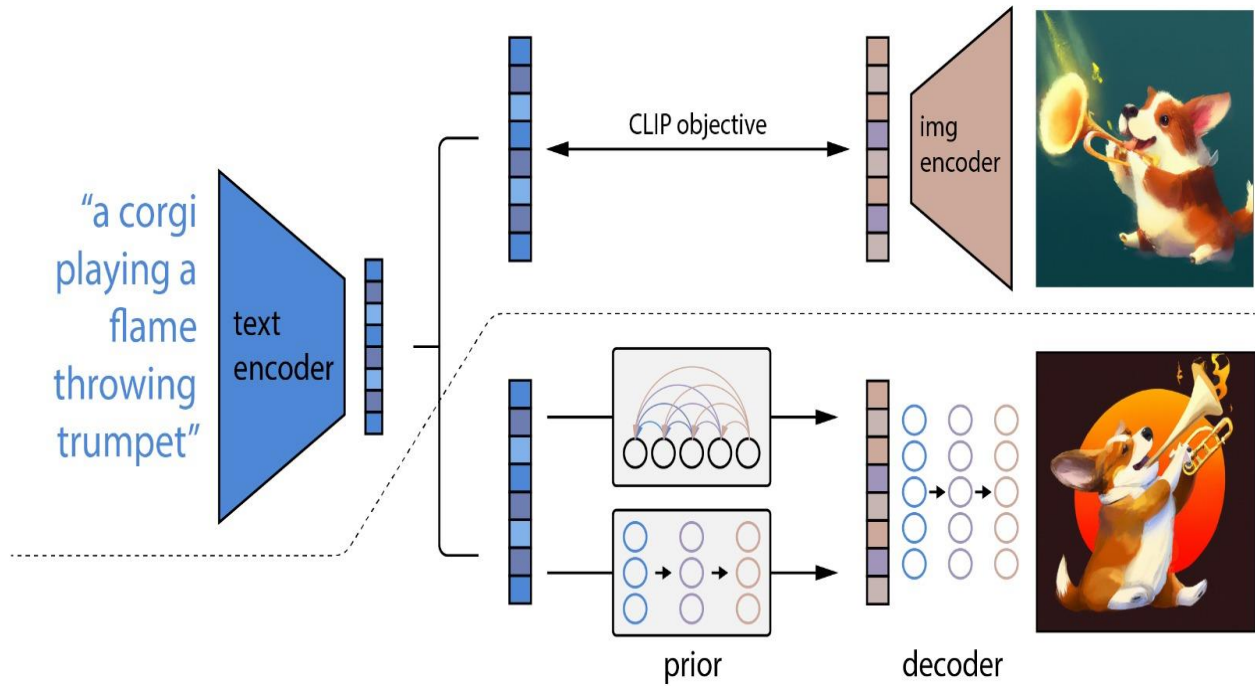


Figure 2.12: unCLIP [11]

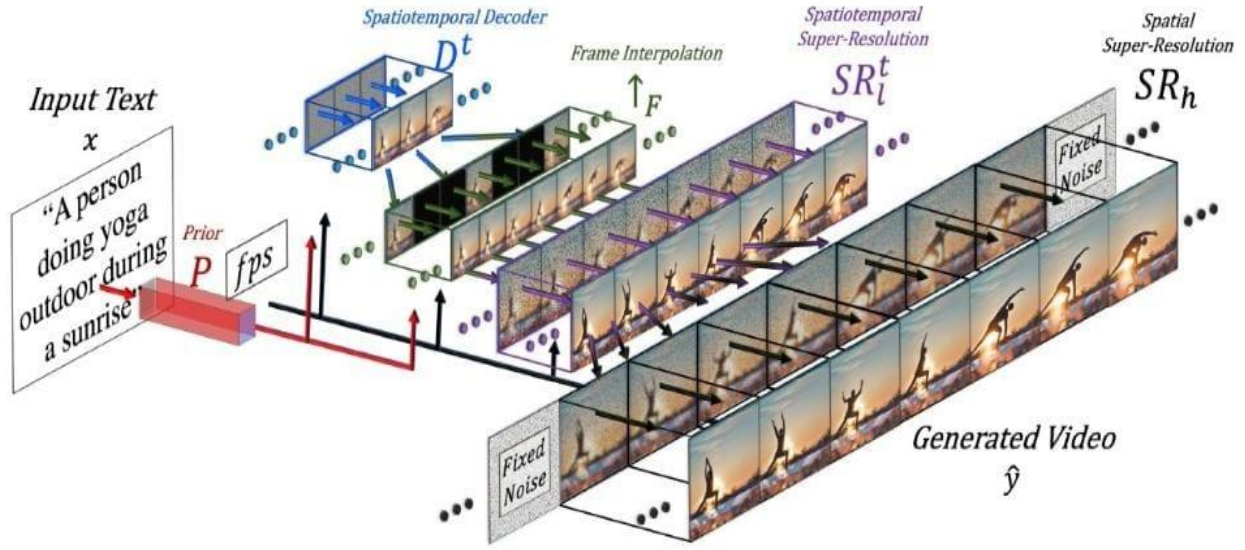
### 2.2.9 Text to Video Generation

It refers to the process of creating a video from a given text description.

One of the tools for text to video generation is Make-A-Video [12]. Figure 2.13 shows its architecture. It consists of three main components:

1. **T2I model:** The following three networks are used to generate high-resolution images from text:
  - A prior network P uses BPE encoded text tokens and text embeddings to generate image embeddings.
  - A decoder network D produces a  $64 \times 64$  RGB image with a poor resolution based on the image embeddings.
  - Two super-resolution networks create the final image by increasing the generated image's resolution to  $256 \times 256$  and  $768 \times 768$  pixels respectively.

2. **Spatio-Temporal layers:** Attention layer and Convolutional layer are modified to accommodate temporal dimension in addition to spatial dimension.
3. **Frame interpolation network:** An extrapolation network and masked frame interpolation are trained to increase the frames. Frame interpolation increases the frames by generating a smooth video while frame extrapolation increase the frames by extending the video length.



**Figure 2.13:** Make-A-Video [12]

## 2.2.10 Text to 3D Generation

One of the tools for text to 3D generation is Magic3D [13]. Figure 2.14 shows its overview. It is a coarse-to-fine framework, divided into two stages, that makes use of effective scene models to enable high-resolution text to 3D synthesis.

1. **Diffusion priors:** Magic3D creates high-resolution textures and geometry in a coarse-to-fine manner by using two separate diffusion priors.

- Base diffusion model is used in the initial phase. The scene model gradients are computed using this diffusion prior through a loss specified on low-resolution  $64 \times 64$  displayed images.
  - Latent diffusion model is used in the second stage, which enables the back-propagation of gradients into displayed images at a high resolution of  $512 \times 512$ .
2. **Scene models:** In order to accommodate the higher quality of produced images for the high-resolution priors input, two separate 3D scene representations are catered to at coarse and fine resolutions with two different diffusion priors.
  3. **Optimization:** It operates first on a rough neural field representation and then on a fine-grained textured mesh.
    - **Neural field optimization:** In order to promote shape growth during the initial stages of optimization, a 256-resolution occupancy grid, with 20 values, is initialized. Every 10 iterations, the grid is changed, and an octree is created to skip empty spaces. The occupancy grid decays by 0.6 with each update. An MLP is used to forecast the normals rather than inferring them from density differences. This aids in lowering the coarse model optimization's computing expense. Using a genuine surface rendering model allows for the precise tuning of optimization and the production of accurate normals. An environment map MLP forecasts RGB colors as a ray direction function. Weighing down the learning rate by  $10\times$  helps the model concentrate more on the geometry of the neural fields.
    - **Mesh optimization:** The density field (coarse) is transformed into a Signed Distance Field (SDF) to optimize a mesh. The optimized color field is then directly initialized in the volume texture field. Using a differentiable rasterizer, high-resolution images are produced during optimization. Focal length is increased when rendering the mesh to close in on object specifics to recover high-frequency information. The angle discrepancies are further regularized between neighboring faces on the mesh to promote the smoothness of the surface.

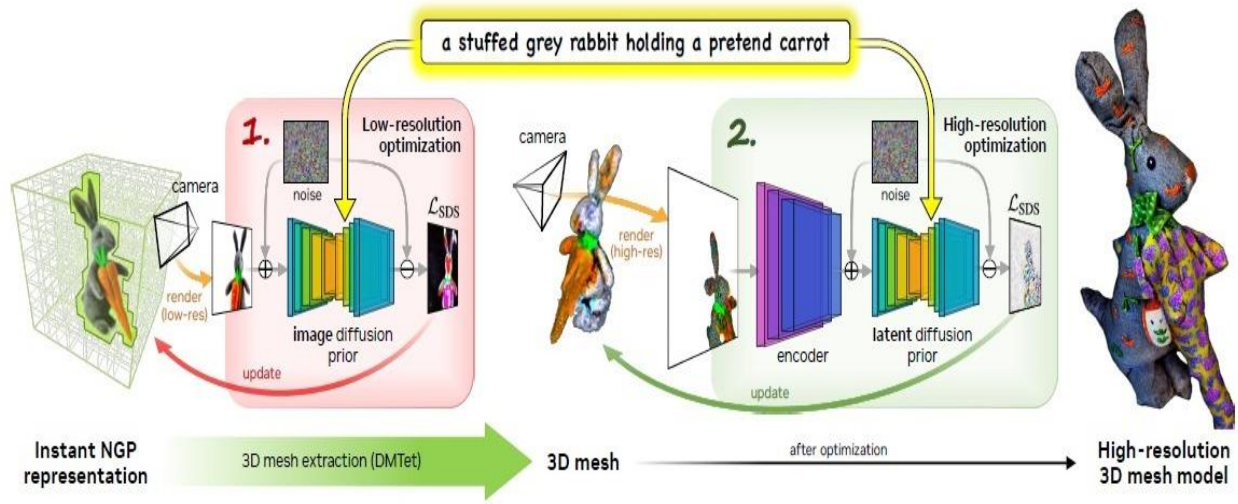


Figure 2.14: Magic3D [13]

## 2.2.11 Text or Speech to Animation

It is the process of creating an animation from text or speech.

### 2.2.11.1 Text/Speech Driven Full-Body Animation

The human body and face are created by two branches, respectively, given a segment of speech and text. Face animation employs a learning-based approach to integrate expressions while body animation synthesizes skeleton motion using database retrieval. Skinning and rendering are then used to create full-body animation [14]. Figure 2.15 shows its overview. The whole framework is divided into two parts: face animation and body animation.

1. **Face animation:** Lower face's lip movement and a variety of upper face expressions make up the two main categories of facial movements.

- **Lip movement generation:** A cross-modal transformer encoder is constructed that uses both textual data and speech to produce precise and synchronized lip movements. By using a temporal alignment analyzer, phoneme alignment annotation is extracted in accordance with speech and textual scripts for textual

information modeling while Mel Frequency Cepstral Coefficients (MFCCs) and Mel Filter Bank (MFB) features are combined for the speech signals. The suggested transformer encoder accepts a series of concatenated audio features and phoneme embedding with a 25 frames window size and a length of 1 second as input. The transformer encoder uses a multi-head self-attention technique across many modalities to efficiently describe the temporal context information.

- **Expression generation:** The motions of the eyebrows and eyes have long-term dependencies on speech rhythm and speaker intention that make up the majority of the upper face's facial expression. The facial expression is divided into two half.
    - Firstly, rhythmic expression movement are generated using a learning-based framework. A transformer decoder is used to forecast the end expression movements based on history motion information and audio. Specifically, the present speech signals are modeled using an audio encoder while the history expressions are modeled using a motion encoder. Structural Similarity Index Measure (SSIM) loss is utilized to examine the structural similarities between the ground truth and predicted expression since synthesizing expression is a one-to-many mapping.
    - Secondly, sentiment analysis is used to extract semantic tags from literary scripts to generate intent-driven face expressions. Semantic tags include words like fear, emphasis, sadness, happiness etc. To create a final varied and expressive facial animation, the created rhythmic expression can be combined with the appropriate intent-driven expression. Finally, the lip movements can also be integrated.
2. **Body animation:** For the given text and speech, a graph-based technique is used to fully automate body motion synthesis. There are two stages of the suggested procedure. On the basis of an existing motion database, a graph is constructed. The motion segments are then extracted from the graph based on the characteristics of the provided speech or text and concatenated into a lengthy sequence.



- **Motion graph construction:** Non-semantic motions and semantic motions are the two basic categories of motions in the database. Obtaining graph nodes and creating graph edges are part of the motion graph creation process. To create the graph nodes, motion strength of long sequence is extracted, and the frame indices of the regional minima are used as dividing points. A graph edge can be formed if there is a sufficiently low transition cost between two neighboring nodes.
- **Graph-based optimization and retrieval:** Given a segment of text or voice, the input is analyzed first, and broken down into numerous phrases based on the text's structure, and then unique semantic text is identified. Using the semantic text and the rhythmic resemblance between the spoken phrase and motion segment, the motion graph is used to extract all meaningful motion segments.

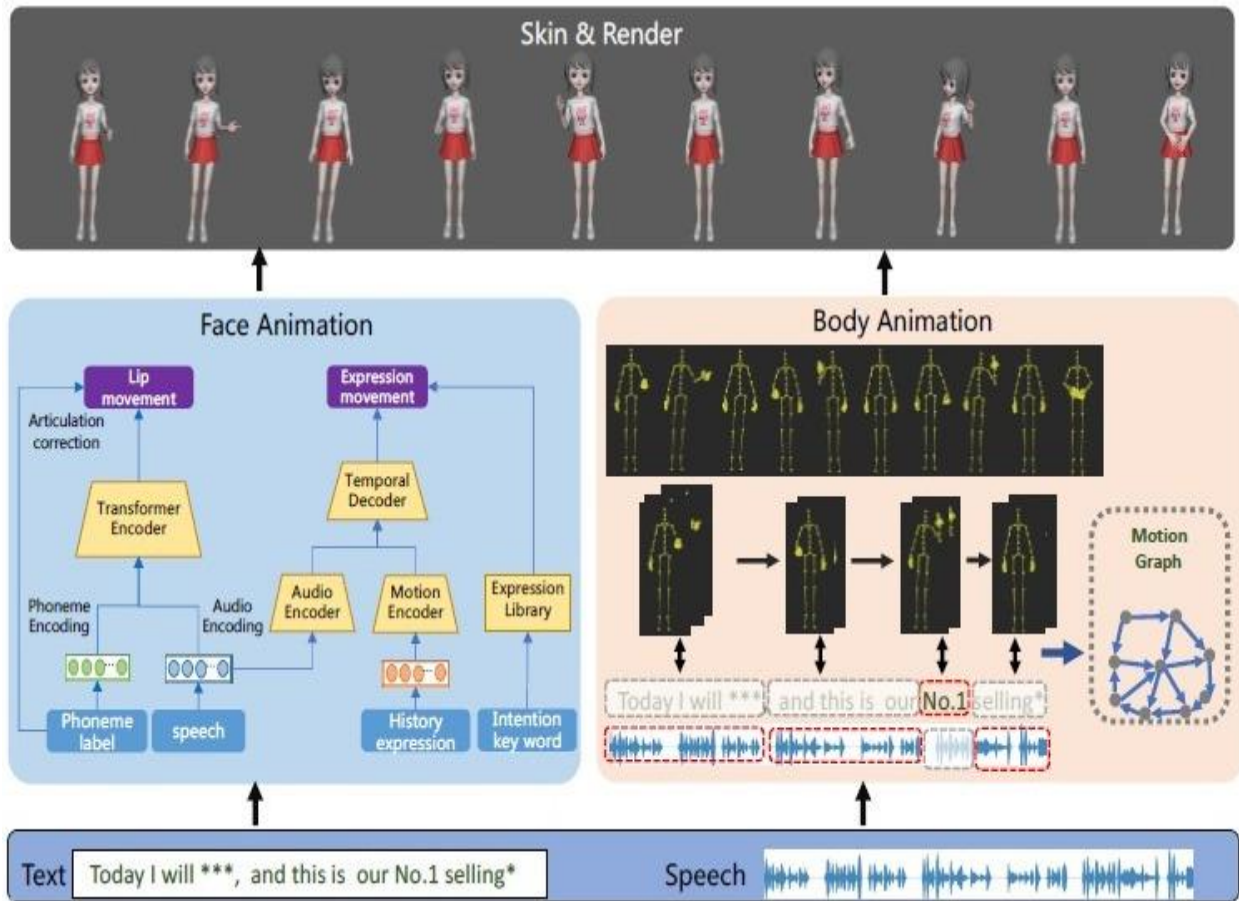
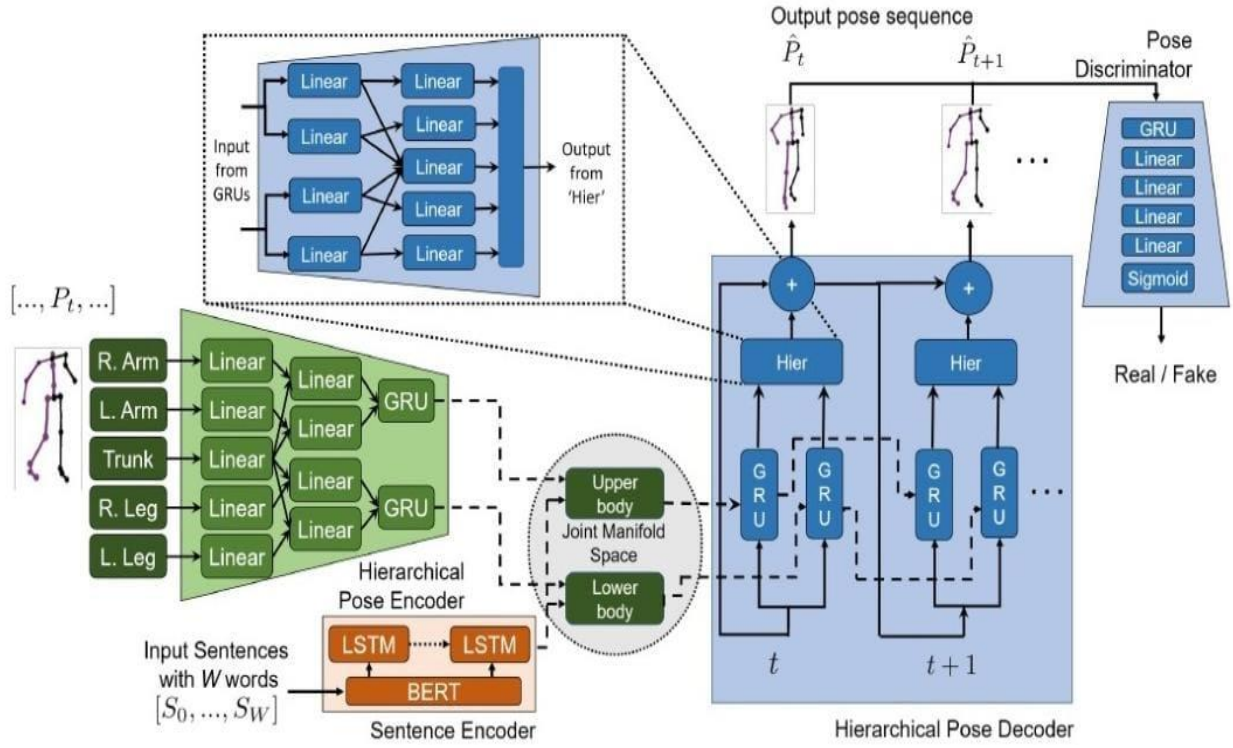


Figure 2.15: Full-Body Animation Framework [14]

### 2.2.11.2 Text Based Compositional Animations

A pose discriminator, a sentence encoder, and a pose autoencoder are used to train the model [15]. The lower and upper body poses and the natural language are jointly learned by the model. Two manifold vectors are created from ground truth pose sequence using two-stream pose encoder. The word embeddings are encoded by two-stream sentence encoder, which creates two latent vectors by mapping them to the latent space. These two manifold vectors are used by two-stream pose decoder to learn how to create postures. Figure 2.16 shows the structure of pose discriminator with hierarchical two-stream model. The three main modules are:

1. **Hierarchical pose encoder:** The pose encoder picks up features from the various parts of the body. The components are then hierarchically merged. The human skeleton is broken down into five main components: right leg, left leg, trunk, right arm, and left arm. These five components are encoded using five linear layers and an output dimension. The trunk representation is combined with right leg, left leg, right arm, and left arm to get combined representations of (trunk, right leg), (trunk, left leg), (trunk, right arm), and (trunk, left arm). These combined representations are then forwarded to additional linear layers. The combined representation of legs with trunk and arms with trunk are encoded separately by two GRUs, resulting in two manifold representations – one for lower body and other for upper body. Two manifold representations can be found in the output of the GRUs.
2. **Sentence encoder:** LSTMs are used by sentence encoder to record a complicated sentence's long-range dependencies. Pre-trained BERT model is used for text input representation. There are 24 levels in total, each of which represents a separate linguistic concept of syntax or semantics.
3. **Hierarchical pose decoder:** Posture decoder can be conceptualized as a collection of hierarchical decoder units, which builds the output pose repeatedly by using the generated pose from the preceding time step. Between the input and output of each individual decoder unit, a residual connection is added. Each decoder unit is made up of two GRUs, several linear layers, and a hierarchically organized system of layers. The pose encoder's hierarchical structure is mirrored in the decoder unit's linear layers.



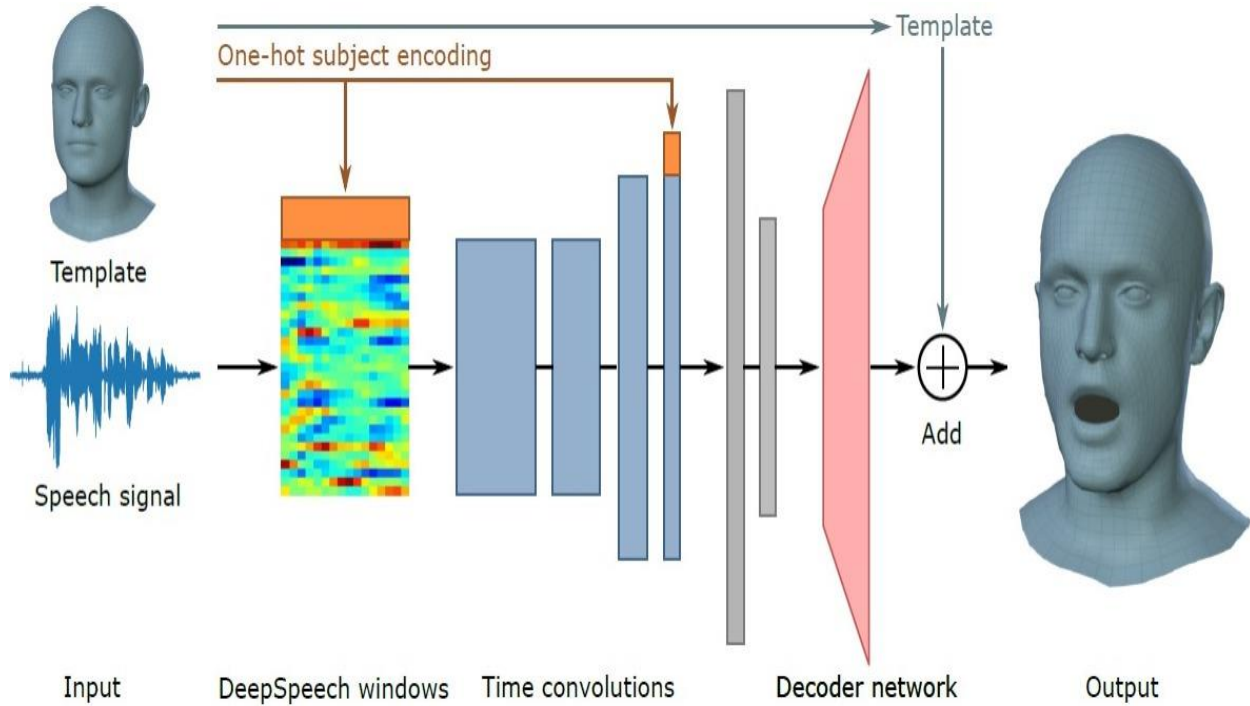
**Figure 2.16:** Pose Discriminator with Hierarchical Two-Stream Model [15]

### 2.2.11.3 Voice Operated Character Animation (VOCA)

The target 3D mesh is the intended output of VOCA [16] which takes raw audio stream and a subject-specific template as inputs. A low-dimensional embedding of audio features is learned by the encoder, and 3D vertex displacements is created by the decoder using this embedding. Figure 2.17 shows the network architecture of VOCA. Its model architecture is as follows:

1. **Encoder:** Two fully linked layers and four convolutional layers make up the encoder. Final convolutional layer and speech features are trained across multiple subjects. They learn subject-specific styles when conditioned on subject labels. Each convolutional layer utilizes a kernel of  $2 \times 1$  stride and  $3 \times 1$  dimension to learn temporal features and minimize the dimensionality of the input. 32 filters are learnt for the first two convolutional layers and 64 filters for the final two in order to keep the number of parameters low and prevent over-fitting. Two completely linked layers follow the final convolutional layer, which is concatenated with the subject encoding.

2. **Decoder:** A fully connected layer with a linear activation function serves as the decoder, producing an array of vertex displacements. Training data's vertex displacements serve as the initialization (with 50 PCA components) for the layer's weights, while zeroes serve as the initialization for the bias.
3. **Animation control:** The output speaking style changes during inference when the eight-dimensional one-hot-vector is changed. A 3D face in 'zero pose' is produced by VOCA.



**Figure 2.17:** VOCA Network Architecture [16]

## 2.3 PERSONALIZED CONTENT ORGANIZATION

The creation of GUIs is one area in which AI is being used. On a whiteboard, designers frequently share concepts and preliminary sketches to begin the user interface design process [17]. The development team manually converts a drawn design into a functional HTML wireframe once it has been captured in a photograph to start the development process. This demands a lot of work and frequently causes delays in the design process.

AI for web development has following challenges:

1. **Hidden complexity demands:** It is arguable that more expertise and knowledge are needed for reliable machine learning/computer vision code development than for conventional software development. For instance, programmers use well-liked frameworks like Keras to create deep neural networks (DNNs). These frameworks conceal the difficulty of creating computational graphs behind practical APIs that enable the development of quite intricate multi-layer topologies with just a few lines of code. Contrary to traditional software, the resulting DNN will typically compute a result, after training, even if the model contains functional faults. As a result, in the event of misbehaviors, developers only receive the final classification score as feedback, whereas the debugging step necessitates working on the complete network architecture.
2. **Need for explainable AI:** Learning process is somewhat deterministic, the sheer complexity of a learning system's internal workings, caused by a plethora of dynamic parameters (such as weights or biases), makes it practically difficult to extract the model. As a result, the taught models are difficult for humans to interpret, comprehend, or explain. The research community should endeavor to develop ML methods that generate models that are easier to understand while yet being highly accurate. Users should be able to comprehend, control, and trust AI-based software thanks to such models.

AI for web development has following opportunities:

1. **Voice-based search:** Due to the emergence of numerous virtual assistants, including Apple Siri, Amazon Alexa, and Google Assistant, voice-based search has recently gained traction. Web designers must consider how voice-based search will develop as the usage of these digital assistants grows. Consequently, voice-activated AI bots will be a key component of this technology's future.
2. **UX & accessibility:** In both research and practice, AI has the potential to be extremely important for UX. For instance, replacement texts for webpages can be created or fixed using picture recognition. CAPTCHA might soon be replaced with facial recognition. Lip

reading can also be used to create video capture or to automatically summarize texts. AI may actively contribute to the right design of websites for those with disabilities, or web accessibility. For persons who are blind, AI can be used to create Braille texts from photos.

## 2.4 INTERACTIVE LEARNING

LLMs can be used to create personalized learning experiences that are tailored to each learner's individual needs and interests. LLMs are categorized into two types:

- **BERT style** (Encoder-Decoder or Encoder-only)
- **GPT style** (Decoder-only)

**Table 2.1:** GPT vs BERT

	<b>GPT</b>	<b>BERT</b>
<b>Training</b>	Autoregressive LMs	Masked LMs
<b>Model Type</b>	Generative	Discriminative
<b>Pretrain Task</b>	Predict next word	Predict masked words
<b>Transformer</b>	Constrained self-attention	Bidirectional self-attention

### 2.4.1 BERT Style LLMs

The BERT [18] framework is divided into fine-tuning and pre-training. Relatively, fine-tuning is less expensive than pre-training. Figure 2.18 show the architecture of BERT.

1. **Pre-training:** Using various pre-training tasks, model training is done on data that is unlabeled. BERT is not pre-trained using conventional right-to-left or left-to-right language model. Instead, two unsupervised tasks are used:
  - **Masked Language Modeling (MLM):** Masking a portion of the input tokens at random to predict those masked tokens helps in training a deep bidirectional

representation. As in a typical LM, the output softmax is fed with the final hidden vectors. 15% tokens are randomly masked in each sequence.

- **Next Sentence Prediction (NSP):** Understanding the relationship between two sentences is the foundation of Natural Language Inference (NLI) and Question Answering (QA), which is not specifically represented by language modelling. Pre-training for a binarized NSP challenge can be easily produced from any monolingual corpus to prepare a model for training that can comprehend sentence relationships.

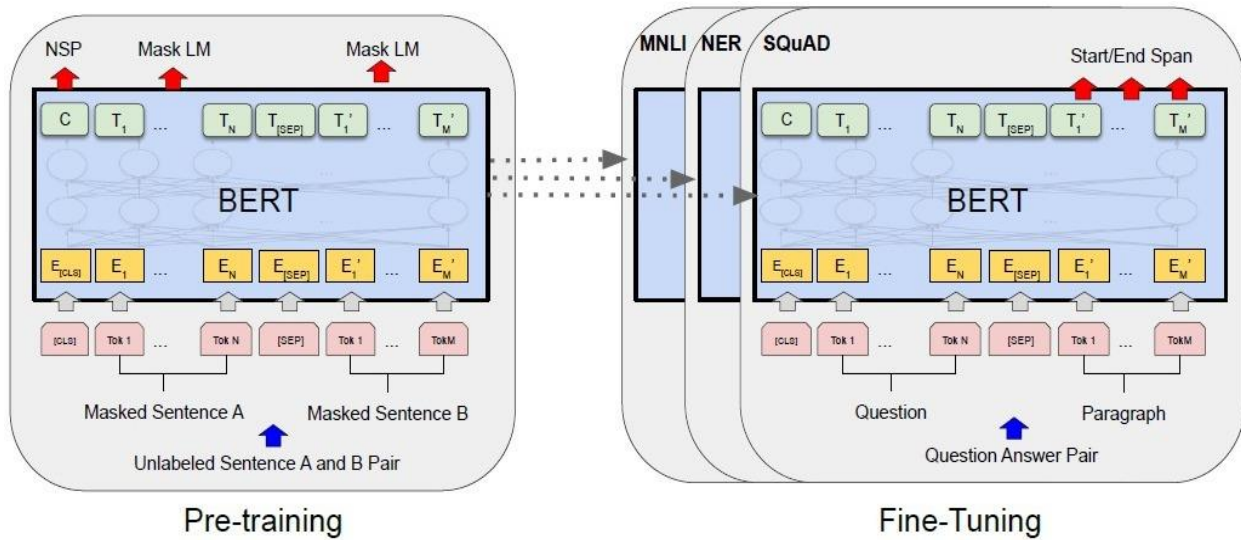


Figure 2.18: BERT's Architecture [18]

2. **Fine-tuning:** Pre-trained parameters are initialized and downstream tasks, with labeled data, is used to fine-tune each parameter. Despite being initialized with same parameters (pre-trained), the downstream tasks have their own fine-tuned models.

- Since the Transformer's self-attention mechanism enables BERT to mimic a variety of downstream tasks, whether they require text pairs or single texts, the relevant inputs and outputs are swapped out.
- Prior to using bidirectional cross attention, it is customary practice for applications containing text pairings to independently encode the text pairs.

- Self-attention mechanism is used to encode a concatenated text pair for incorporating cross attention (bidirectional) between two phrases.

**Table 2.2:** Fine Tuning

At the input	At the output
<p>Sentences A and B from the pre-training are comparable to:</p> <ol style="list-style-type: none"> <li>1. Pairs of sentences in paraphrasing,</li> <li>2. Pairs of premise-hypothesis in entailment,</li> <li>3. Pairs of passage-question in question answering, and</li> <li>4. A text-Ø pair that has degenerated in sequence tagging or text categorization</li> </ol>	<p>For token-level tasks like question answering or sequence tagging, an output layer is fed with token representations,</p> <p style="text-align: center;">and</p> <p>For classification tasks like sentiment analysis or entailment, an output layer is fed with [CLS] representation.</p>

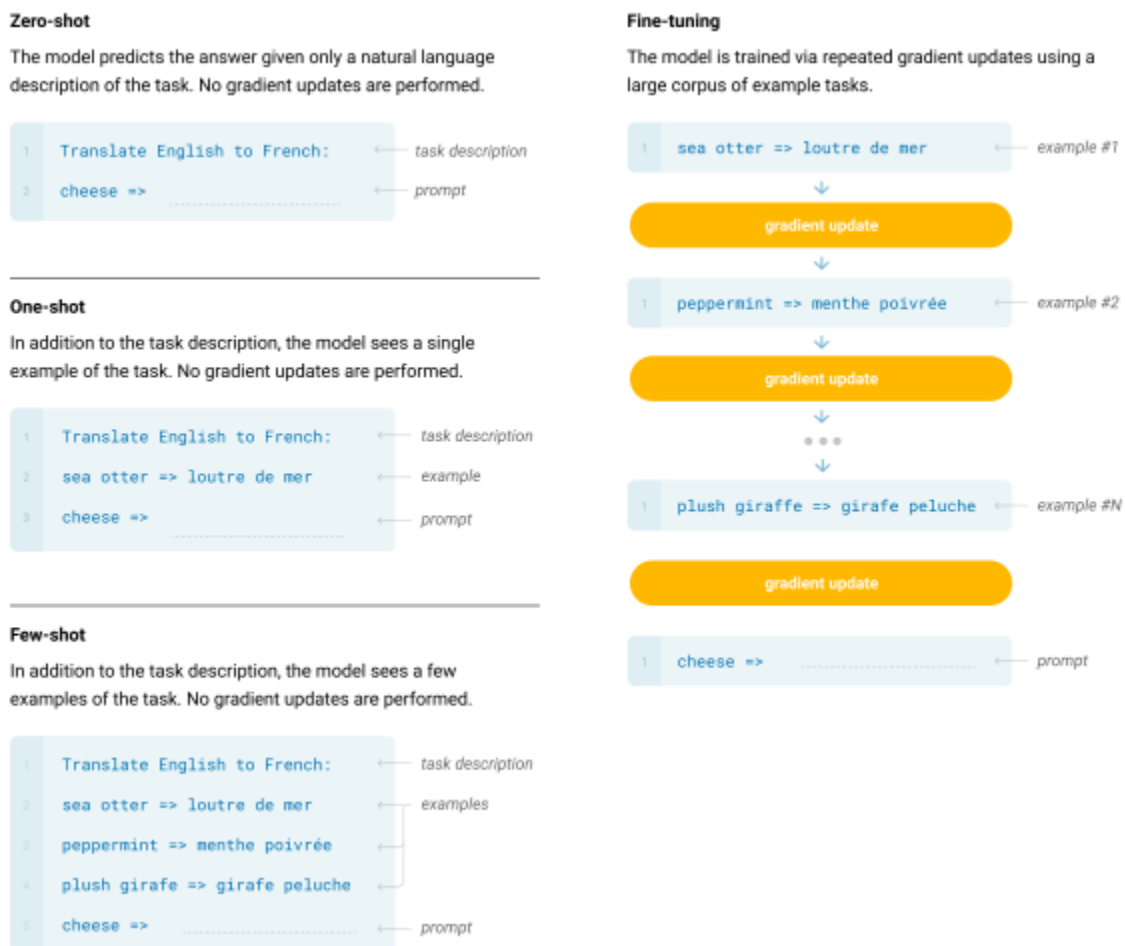
## 2.4.2 GPT Style LLMs

The GPT-3 [19] framework focuses on four spectrums:

1. **Fine-tuning:** It is the most often used strategy in recent years.
  - It comprises of training on a supervised dataset tailored to the intended task in order to update the weights of a model that has been pre-trained.
  - Strong performance on numerous benchmarks is the key benefit of fine-tuning.
  - The key drawbacks include the requirement for a brand-new, sizable dataset for each job, the possibility of subpar generalization outside of distribution, and the potential for exploiting fictitious aspects of the training data.
2. **Few-shot:** The model is trained by giving it a few examples of the problem at inference time, but weight updates are not permitted.



- It provides the completion when K examples of completion and context are given, followed by one last example of context.
- Few-shot's principal benefits include a significant decrease in the need for task-specific data and a decreased risk of learning an excessively narrow distribution.
- The key drawback is that this strategy has so far produced substantially worse results than cutting-edge fine-tuned models. Additionally, some task-specific data is still needed.



**Figure 2.19:** GPT-3 [19]

3. **One-shot:** It is identical to few-shot with the exception that just one demonstration is permitted and additionally a natural language task description is allowed.

4. **Zero-shot:** The only differences between zero-shot and one-shot are that no demos are permitted and only a natural language description of the assignment is given to the model.

### 2.4.3 Chat-REC

Recommender systems [20] give suggestions to users based on their preferences and actions. NLP methods have shown to be useful in broadening the application of recommender systems outside the realm of conventional user data. Figure 2.20 shows an overview of Chat-REC.

LLMs and recommender systems can be bridged together to write personalized prompts and make informed decisions. User-item historical interactions, user profile, user query and dialogue history are some of the inputs that can be entered in prompt constructor module. There are three types of recommendations:

1. **Candidate set compression recommendations:** The candidate set is reduced using LLMs to enhance recommender system performance.
  - Previous interactions and user's profiles are converted into prompts (including the user rating and item description).
  - LLMs compile user preferences for various products inside a domain, using the above data.
  - LLMs, by utilizing in-context learning, can improve their reasoning skills for suggestions, leading to more precise and individualized product recommendations.
  - This strategy makes sure the user is shown a more focused set of items, improving the chance that they will find something they like.
2. **Cold-start recommendations:** LLMs help in recommending products with little or no user engagement.

- External information is provided to create and cache corresponding embedding representations for new items.
- Similarities between preferences and user requests embeddings and embeddings of item is calculated when presented with new item recommendations to retrieve the most pertinent item information.

3. **Cross-domain recommendations:** LLMs have a thorough understanding of the items in many domains, and they are aware of the relationships between various products in other domains as well. LLMs capacity to transfer customer preferences from one item to another results in recommendations across domains. The scope and relevancy of recommender systems could be significantly increased by this cross-domain recommendation capabilities.

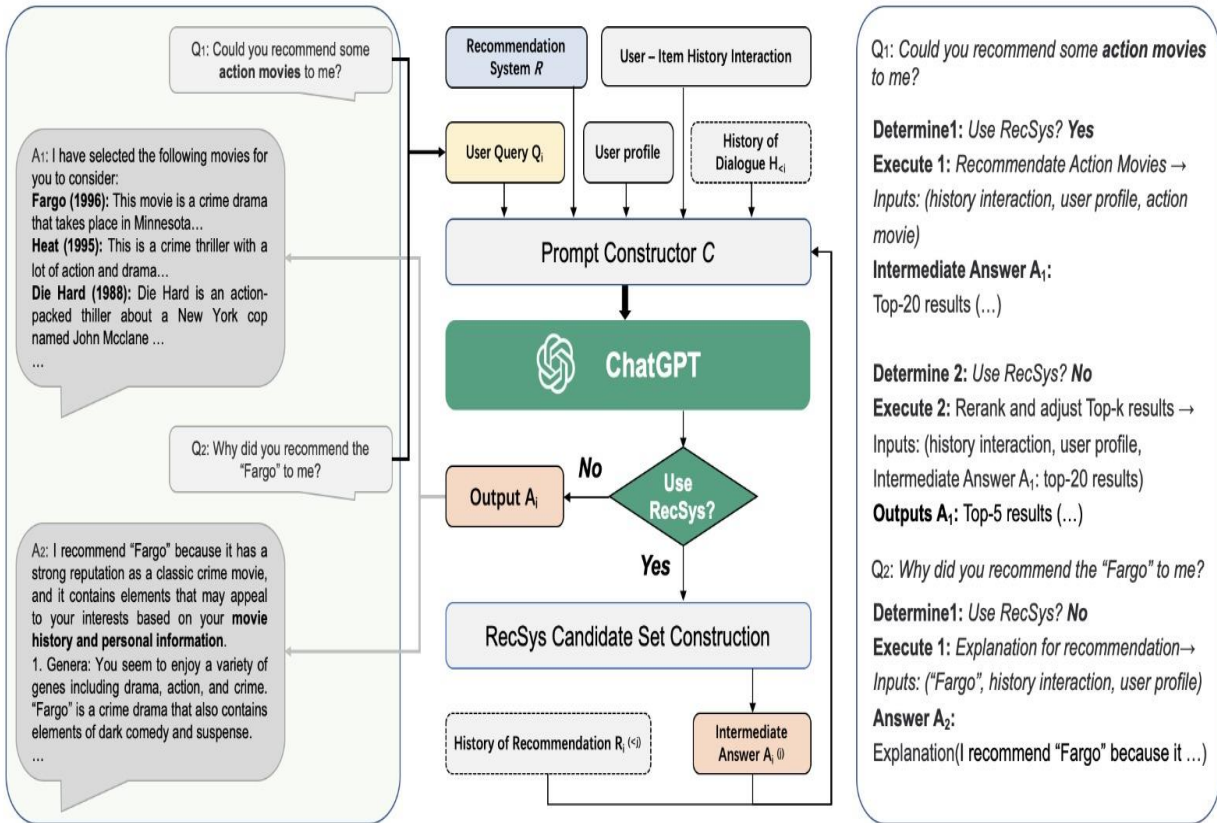


Figure 2.20: Overview of Chat-REC [20]

## 2.5 HAPTICS ENHANCED LEARNING

Research in the areas of virtual reality, mechanics, and haptic perception aims to develop haptic interfaces that let users interact with and feel believable virtual things. The sense of touch in humans is highly developed and capable of perceiving a variety of tactile stimuli [21].

The study of haptic perception aims to comprehend how the human brain interprets touch data. This entails comprehending the skin's spatial and temporal acuity as well as the intricate interplay between various touch sub-modalities like texture, vibration, and pressure.

The physical foundations for the transmission of tactile impulses are the focus of haptic mechanics. Understanding the mechanical characteristics of the skin in addition to the deformations and forces produced when things interact with the skin are all part of this.

The creation of new materials and tools that can be utilized to make haptic interfaces is the focus of haptic feedback. This comprises the creation of actuators that can provide accurate tactile stimuli as well as the creation of haptic displays for efficient and comfortable transmission of these sensations to the skin.

Haptic feedback is useful for a wide variety of virtual reality applications:

1. **Education:** Students can explore and learn about various topics through interactive learning experiences that include haptic feedback.
2. **Entertainment:** Gaming experiences can be made more realistic and engaging by utilizing haptic feedback.
3. **Healthcare:** Therapy for people with disabilities can be given via haptic feedback.

## 2.6 LIMITATIONS

1. There is no integrated platform to use the emerging technologies to let the user create personalized content and learn interactively.
2. There is no integration of virtual reality for content presentation.
3. The majority of the LLMs accept text as input and answer the user's queries in textual form only. So far only GPT-4 has succeeded in taking input in the form of text as well as image and answer user's queries in textual form only. So there is no visual teacher bot to answer learner's questions.
4. Current LLMs suffer from lack of understanding context, biased training data, lack of explainability, high computational cost, easily fooled by adversarial examples, and sometimes generate inaccurate or harmful content.

## **CHAPTER-3**

### **NEW INITIATIVES**

In this chapter I conceptualized and developed the idea of an integrated virtual scene that will facilitate to let anyone learn from her/his choice virtual human agent and seek answers for her/his questions from the teacher bot in the voice of the teacher. This has been done in the following steps:

1. I created animated content using MNIST dataset on CNN in the blender software, made its video, and embedded this video content in Unity 3D virtual platform.
2. Imported a character with 3 animations (idle, listening and talking) from Mixamo; embedded it in the Unity 3D virtual platform; and added new functionalities to answer learner's questions in a sequential manner.
3. If the learner is not satisfied with the answers by the Teacher\_Bot then (s)he can click the 'Connect' button to discuss with a friend or a teacher or anyone.

### **3.1 CONTENT CREATION**

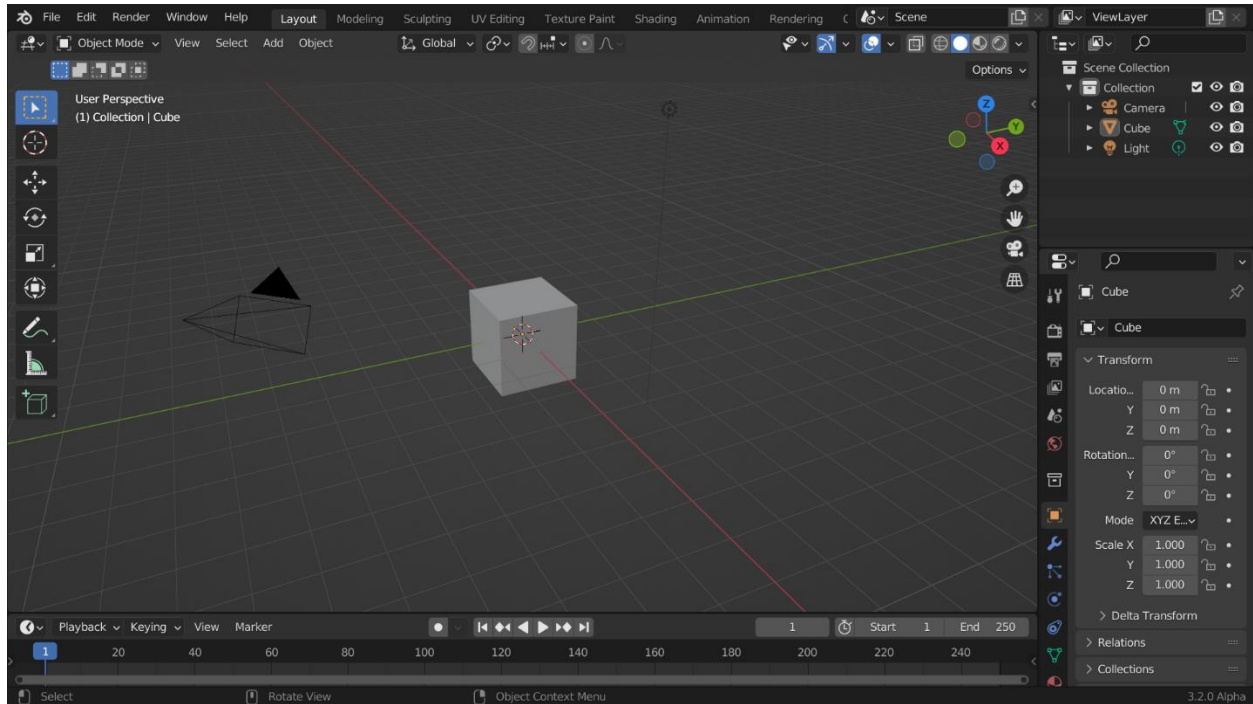
Content is created in the form of animation for image classification using Convolutional Neural Network (CNN). The dataset used for this purpose is famously known as MNIST handwritten digit dataset (for classification problems).

The MNIST dataset is trained using Multiclass (or Multinomial) logistic regression. To accommodate multi-class classification problems natively, the multinomial logistic regression technique extends the logistic regression model by switching to cross-entropy loss from loss function and forecast a multinomial probability distribution with a probability distribution. It takes  $10 \times 10$  image of a handwritten digit as an input and produces 10 probabilities as an output.

Blender's interface consists of following three parts:

1. **Topbar:** Used for rendering, configuring settings, exporting files, saving, and importing.
2. **Areas:** Serves as the main workspace.
3. **Status bar:** It displays suggested shortcuts and pertinent facts.

Figure 3.1 shows the default Blender window.

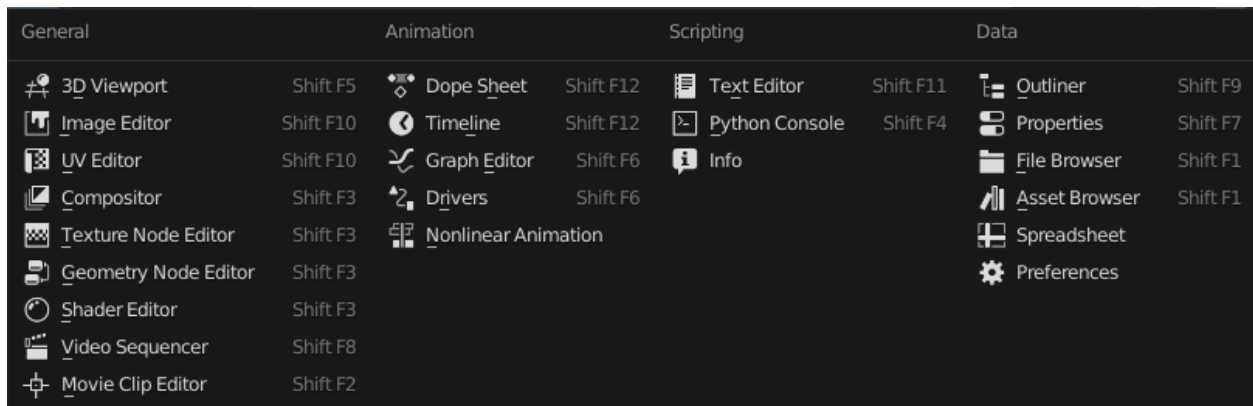


**Figure 3.1:** Default Blender Window

We now open the text editor and create a new text file for our script.

- For viewing and altering various parts of data, Blender offers a variety of editors. An area encloses an Editor and controls the size and positioning of the Editor within the Blender window.
- We can switch the Editor by using the Editor Type selector.

Figure 3.2 shows the Editor Type selector.



**Figure 3.2:** Editor Type Selector

The script starts with the import of three packages, namely:

- bpy: for executing blender command in python
- vector from mathutils (Math Types and Utilities): for visualizing weights
- numpy: to load all the files we need

We now load the three files (inputs, outputs and weights), that were earlier saved while training the MNIST dataset using Multinomial logistic regression, using numpy package. Figures 3.3., 3.4, 3.5, 3.6, 3.7, and 3.8 show the script used for creating content in Blender software.

```
import bpy
from mathutils import Vector
import numpy as np

#10x10x128
inputs = np.load("/Users/inputs.npy")
#1x10x128
outputs = np.load("/Users/outputs.npy")
#100x10x128
weights = np.load("/Users/weights.npy")
```

**Figure 3.3:** Import Packages and Load Files



```

def set_material_colors(obj, values, frame_num):
    # Create a new material for the object
    mat = bpy.data.materials.new(name="ColorMat")
    obj.data.materials.append(mat)

    # Loop through the values and set the material color for each one
    for i, val in enumerate(values):
        # Create a new color and set it on the material
        color = (val, val, val, 1.0)
        mat.diffuse_color = color

        # Insert a keyframe for the material color every frame_num frames
        frame = i * frame_num
        mat.keyframe_insert(data_path="diffuse_color", frame=frame)

```

**Figure 3.4:** Material Colors Function

```

def create_cube_grid(n, colours, spacing, size, y_position=0):
    cube_locations = []
    for i in range(n[0]):
        for j in range(n[1]):
            x = j * (spacing + size) - (n[1] - 1) * (spacing + size) / 2
            y = y_position
            z = (n[0] - 1) * (spacing + size) / 2 - i * (spacing + size)

            # Create a new cube
            bpy.ops.mesh.primitive_cube_add(location=(x, y, z))
            cube = bpy.context.active_object
            cube.scale = (size, size, size)

            cube_locations.append(Vector((x, y, z, 0)))
            set_material_colors(cube, colours[i][j], 20)
    return cube_locations

```

**Figure 3.5:** Cube Grid Function

```

def create_curve_object(point_a, point_b, colours, thickness):
    # Create a new curve
    curve_data = bpy.data.curves.new('Curve', 'CURVE')
    curve_data.dimensions = '3D'

    # Create a new spline
    spline = curve_data.splines.new('POLY')
    spline.points.add(1)
    spline.points[0].co = point_a
    spline.points[1].co = point_b

    # Set the thickness of the curve
    curve_data.bevel_depth = thickness

    # Create a new object and link it to the scene
    obj = bpy.data.objects.new('Curve', curve_data)
    bpy.context.scene.collection.objects.link(obj)

    # Set the object to use the curve as its data
    obj.data = curve_data
    set_material_colors(obj, colours, 20)

    return obj

```

Figure 3.6: Curve Object Function

```

def create_curves_between_points(point_list_1, point_list_2, colours, thickness):
    # Loop through all combinations of points in the two lists
    for i, point_a in enumerate(point_list_1):
        for j, point_b in enumerate(point_list_2):
            # Create a curve object between each pair of points
            create_curve_object(point_a, point_b, colours[i][j], thickness)

```

Figure 3.7: Curves Between Points Function

```

l2 = create_cube_grid((10, 10), inputs, 1, 0.5, y_position=-10)
l1 = create_cube_grid((1, 10), outputs, 1, 0.5, y_position=10)

#weights should be 100x10x128
create_curves_between_points(l2, l1, weights, 0.005)

```

Figure 3.8: Weights

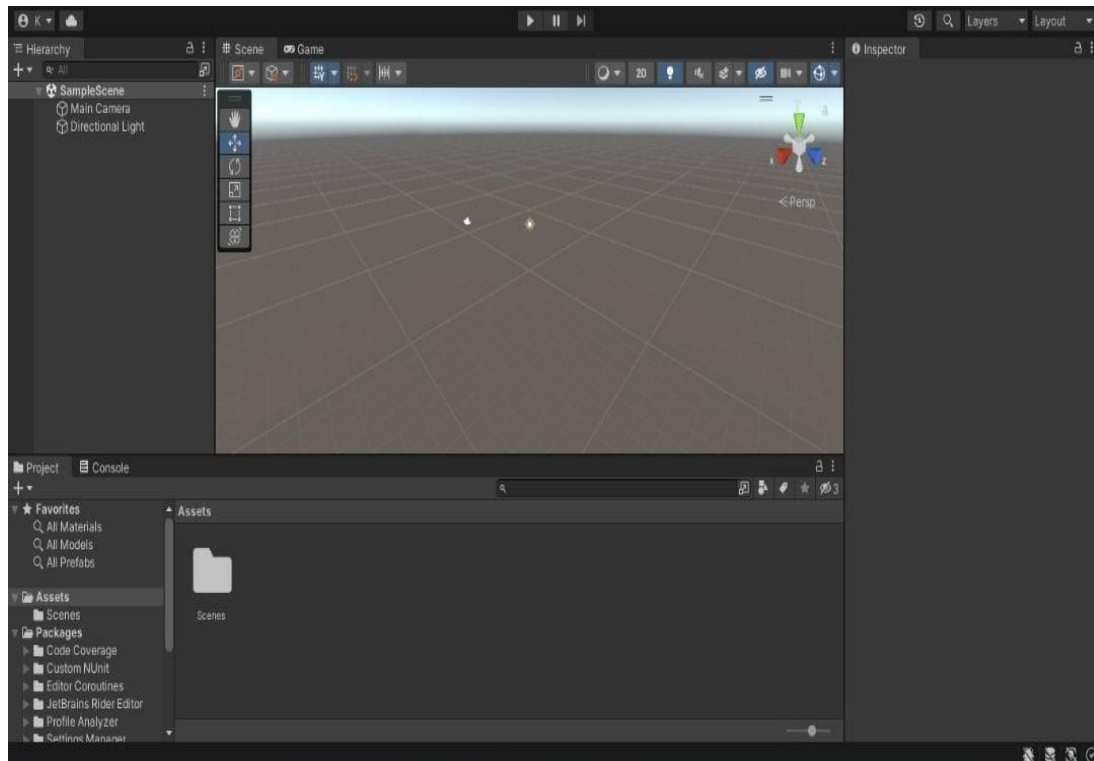
## 3.2 STRUCTURE AND FUNCTIONING OF INTEGRATED VIRTUAL SCENE

The integrated virtual scene consists of three parts:

1. A brief overview of Unity 3D virtual platform.
2. Embedding the Teacher Bot in the Unity 3D virtual platform
3. Embedding the video content (created in section 3.1) in the Unity 3D virtual platform.

### 3.2.1 Unity 3D Virtual Platform

Unity Technologies created a cross-platform gaming engine known as Unity. It is used to make interactive simulations as well as both 3D and 2D games. It is one of the most widely used game engines worldwide. Figure 3.9 shows the default Unity window.



**Figure 3.9:** Default Unity Window

Unity's interface consists of eight parts: toolbar, hierarchy window, game view, scene view, overlays, inspector window, project window, and status bar.

## 1. Toolbar

- It can be found in the Unity Editor's header.
- It is the only area of the Unity interface that cannot be rearranged.
- It consists of seven different parts that are described in table 3.1.

**Table 3.1:** Toolbar Control Tools

Control	Description
Account	Account drop-down menu allows us to access our Unity account.
Manage Services	Unity services window will appear when clicked on the Cloud button.
Play, Pause and Stop	Utilizing the Game view's Play, Pause, and Stop buttons.
Undo History	It is used to inspect, undo, or redo activities done in the Editor.
Global Search	Search is used to conduct internal Unity searches and take action on the results.
Layers	Layers drop-down menu lets us choose which objects will be visible in Scene view. With the aid of layers, we can divide GameObjects in our scenes. To change how GameObjects in our scene connect with one another, we can use layers with scripts and through the UI.
Layout	It lets us to organize our views differently, save it or select an existing one from the Layout drop-down option to load. When using tabbed views, the top of the "active" tab (the one with focus) always displays a thin blue stripe.

## 2. Hierarchy window

- Every GameObject in a Scene, including models, cameras, and prefabs is visible in the Hierarchy window.

- GameObjects used in a Scene can be sorted and grouped using the Hierarchy window.
- GameObjects are also added to or removed from the Hierarchy window when they are added or removed from the Scene view.
- Other Scenes, each holding its own GameObjects, can also be found in the Hierarchy window.
- GameObjects are organized into groups using parent-child hierarchies, also known as parenting.
- Other GameObjects that inherit an object's properties may be present.
- In order to transform, scale, or move a group of GameObjects, we can link GameObjects together.
- All child GameObjects are moved together with the parent GameObject whenever the parent GameObject is moved.
- Additionally, hierarchical parent-child GameObjects are a possibility. The root GameObject, or original parent GameObject, still remains the source of all nested objects.
- The rotation and movement of the parent GameObject are inherited by the child GameObjects.

### **3. Game view**

- Cameras are used to render the Game view.
- Game view displays how the completed, created application looks.
- To manage the player's view, use of one or more cameras is needed.
- Unity lets us switch between the Simulator view and the Game view in Unity.
- The Simulator view displays how our developed application appears on a mobile platform.
- We can run our project in the Play mode to see how it functions just like a finished application.
- Play mode can be controlled through the buttons on the Toolbar.

- Any modifications made while being in Play mode are just temporary and are undone when we exit it.
- Unity darkens portions of the UI outside of the Game view when we switch to Play mode.

#### 4. Scene view

- We can interact and visualize with the world designed in the editor.
- We can choose, move, and alter GameObjects that serve as scenery, lights, cameras, characters, and other things in the scene view. Scene view tools are described in table 3.2.

**Table 3.2:** Scene View Tools

Topic	Description
Position GameObjects	GameObject's transform values can be modified.
Scene view navigation	Efficiently navigate the scene view.
Scene view camera	Set the scene camera as desired.
Gizmos menu	Show and hide gizmos.

- **Position GameObjects:** We can adjust any Gizmo axis or type numbers directly into the transform component's number fields in the Inspector to modify the GameObject's transform component. In the scene view, there are five built-in gizmos or transform modes: Move, Rotate, Scale, RectTransform, and Transform.
- **Scene view navigation:** There are a few navigation controls available in the scene view to let us move around efficiently:
  - **Scene gizmo:** We can adjust the viewing angle and projection mode while also seeing the current orientation of the scene view camera. A conical arm is present in the scene gizmo. The front arms are identified as X, Y, and Z. We can align the scene view camera with any conical axis (such as front view, left view, and top view) by simply clicking on one of its arms. If we

choose the **free** keyword, the viewing angle is restored to default settings. Additionally, we can turn **perspective** off and on. By doing so, we can switch the scene view's projection mode between orthographic and perspective.

- **Zoom, Move and Orbit tools:** The three main navigational actions in scene view are zooming, orbiting, and moving. For optimum accessibility, Unity offers a variety of ways to carry them out. **Arrow keys** are used to traverse the Scene, like 'walking' through. The camera can be moved backward and forward in the direction it is facing by using the down and up arrow buttons. The sideways view is panned with the right and left arrow keys. **View tool** has three mouse controls, namely pan, orbit, and zoom that are accessible when view tool is chosen. **Flythrough mode** is used to get around the scene view. Perspective mode is intended for flythrough mode while our view in Orthographic mode revolves around the camera.
- **Scene view camera:** Options for setting up the scene view camera can be found in the camera settings menu. The settings on GameObjects that contain camera components are unaffected by these changes.
- **Gizmos menu:** Gizmos and Icons are present in both the game view and the scene view.
  - **Gizmos:** Visuals connected to GameObjects in the scene. In contrast to bitmap images, gizmos are typically wireframes that can be interactive. Directional light gizmo and main camera gizmo are two main built-in gizmos. Gizmos like directional light are passive graphical overlays, i.e., they just display the light's direction. AudioSource is one of the prime example of interactive gizmo as we can change the maximum range by simply clicking and dragging the AudioSource. Additionally, transform, rotate, scale, and move are some more examples of interactive gizmos. Using a script, we can also make our own gizmos.
  - **Icons:** Billboard-style, flat overlays that we can use to clearly show the position of a GameObject. Icons can be displayed in either the scene view

or the game view. Directional light icon and main camera icon are two main built-in icons; we can also attach our own to specific scripts or GameObjects.

## 5. Overlays

- Overlays are persistent, resizable panels and toolbars that can be used as authoring tools in the scene view.
- Additionally, relevant information regarding our choice can be displayed using overlays.
- We can position overlays, decide which overlays to show or conceal, and save, import, or export overlay configurations to enhance our workflow.
- We can adjust our overlay setups and choose which overlays appear in the scene view from the overlays menu. Overlays tools are described in table 3.3.

**Table 3.3:** Overlays Tools

Overlay	Description
Tools	There are six tools available in the tools overlay, namely: view, move, rotate, scale, rect and transform.
Tool settings	To see the settings that are available for the chosen tool, we use the tool settings overlay. For instance, if we use the transform tool, we may choose the handles and placement of any transform tool gizmos in the tool settings overlay.
Grid and Snap	GameObjects' ability to snap to the grid can be controlled using the grid and snap overlay.
View options	For the scene view, view options overlay can be used to choose view options, manage lighting, and manage audio. Only the scene view is impacted by these controls; the created scene is unaffected.
Orientation	Scene camera's orientation can be changed, along with the projection mode and its viewing angle, using the orientation overlay.



## 6. Inspector window

- Nearly all elements in the Unity editor, including GameObjects, materials, assets, components, preferences and in-editor settings, can have their properties and settings viewed and modified via the inspector window.
- **Inspect items:** Depending on what we choose, the inspector window's contents can be viewed and modified.
  - Inspecting **GameObjects**: The inspector shows all of a GameObject's components and materials properties when we choose a GameObject.
  - Inspect **assets**: The inspector shows options that affect how an asset is imported and used by Unity at runtime when we pick an asset. The settings vary depending on the type of asset.
  - Inspect **prefabs**: The inspector window offers several extra choices and displays some additional options when we deal with prefabs.

## 7. Project window

- We can navigate and locate project files and assets in our application primarily using the project window, which shows all of the files associated with our project.
- The project's folder structure is displayed as a hierarchical list in the browser's left panel.
- Unity displays a folder's contents on the right pane when we choose a folder from the list.
- We can collapse or expand the folder by clicking the tiny triangle, revealing any nested folders it may contain.
- The right hand panel displays individual assets as icons that denote their type (such as material, script, or sub-folder). Figure 3.4 shows the project window toolbar.

**Table 3.4:** Project Window Toolbar

Property	Description
Create menu	Lists the assets and any subfolders that can be added to the folder we have selected.
Search bar	Within our project, we can search a file through this search bar.
Open in Search	Enables the Unity Search so that we can refine our search.
Type Search	By choosing this option we can limit our search to a certain kind, such as a Scene, Mesh, or Prefab.
Label Search	To select a tag to search within.

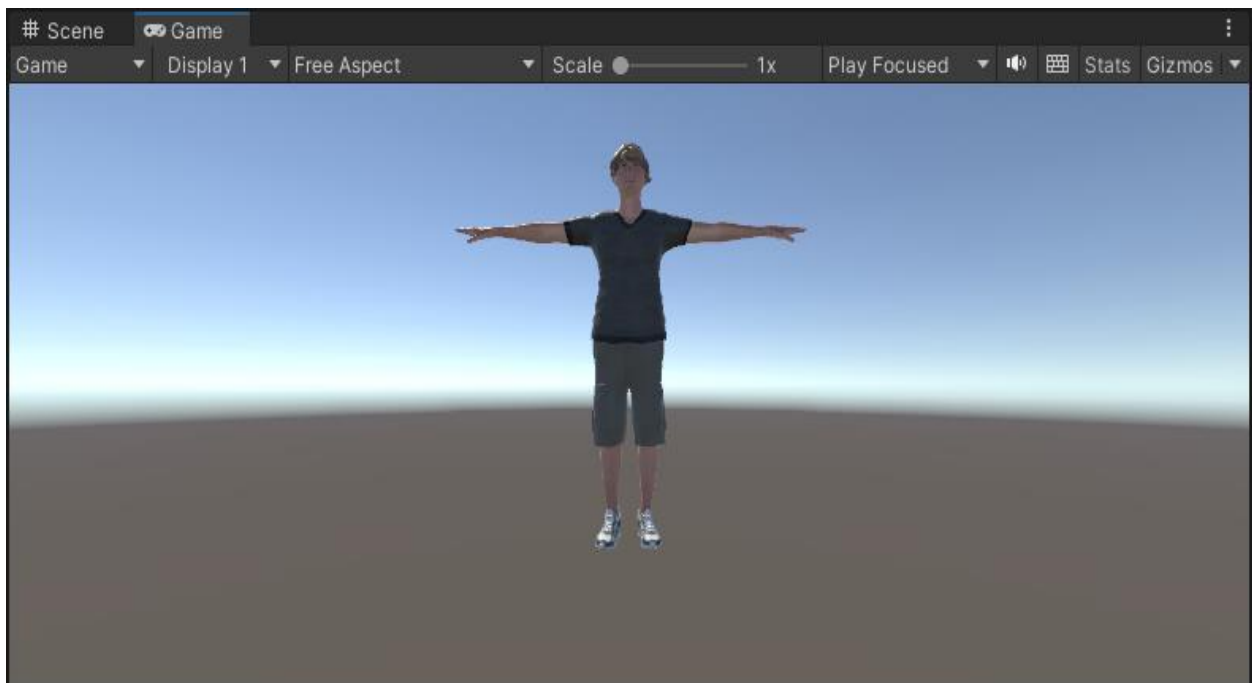
## 8. Status bar

- The status bar offers rapid access to relevant tools and settings as well as notifies about various Unity processes.
- The status bar cannot be moved or rearranged.

### 3.2.2 Embedding the Teacher Bot in the Unity 3D Virtual Platform

1. When we create a new 3D project, a default scene of Unity's interface opens with all the eight parts at their respective places.
2. We create a folder named 'Animations' inside the assets folder. This assets folder lies inside the project window.
3. We download a character and its associated animations (idle, talking and head nod) from Mixamo platform. Mixamo is an open-source technology company that gives free access to various characters and mocap (motion capture) animations. We can also modify our animations on Mixamo according to our style.
4. We import the .fbx character file and .fbx animations file and place it inside the 'Animations' folder. We name the character as Teacher\_Bot.

5. We copy the Teacher\_Bot character from 'Animations' folder and paste it inside the hierarchy window. In the hierarchy window, the Teacher\_Bot is accompanied by its two siblings that are directional light and main camera.
6. Now, the Teacher\_Bot character is visible in the game view with a T-pose. At this moment, the character is without any textures and materials.
7. We create two new folders, namely textures and materials respectively, inside the 'Animations' folder. We select the Teacher\_Bot character inside the 'Animations' folder and in the inspector window we click on materials option to extract the textures and materials one by one. This leads to a visible change, i.e., the Teacher\_Bot standing in the game view is now covered with some clothes and accompanying accessory, if any. Figure 3.10 shows the Teacher\_Bot in game view.



**Figure 3.10:** Teacher Bot in Game View

8. Till this point, the Teacher\_Bot character is not able to play any animation, so we need to change some settings to make the character play the different animations.
9. We select the Teacher\_Bot character from the 'Animations' folder and in the inspector window we can see there are four options that are visible, namely model, rig, animation,

and materials. We click on rig option to change the animation type from ‘legacy’ to ‘humanoid’ and choose the avatar definition as ‘create from this model’.

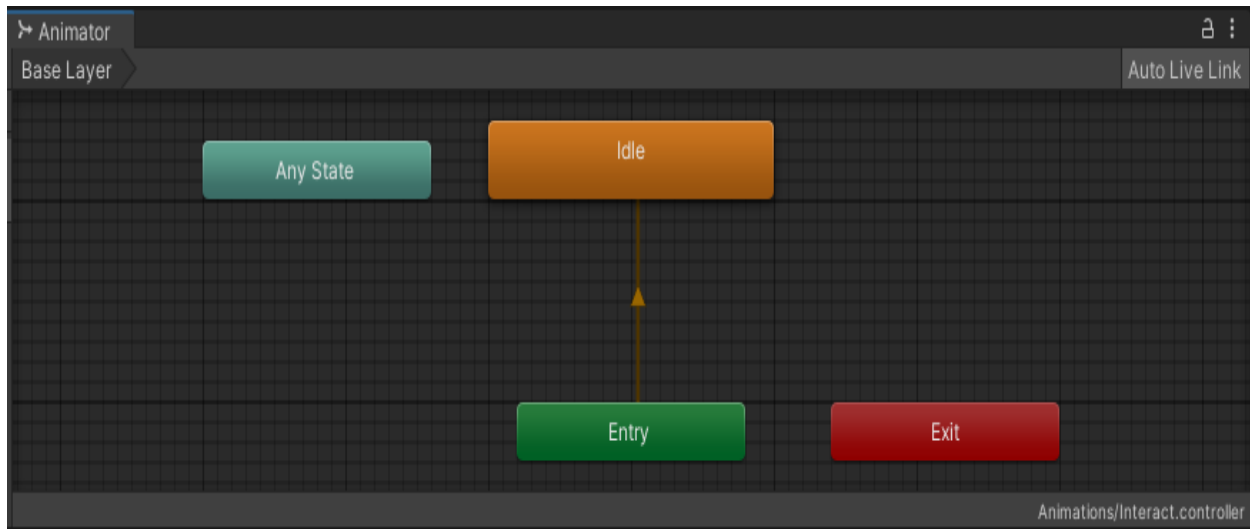
10. This prompts a change in the inspector window. Now, when we select the Teacher\_Bot character in the hierarchy window we can see an animator component below the transform component of the Teacher\_Bot character. The animator component consists of five options, namely controller, avatar, apply root motion (check box), update mode, and culling mode.
11. The default values set in the animator component for ‘Teacher\_BotAvatar’ are described in table 3.5.

**Table 3.5:** Animator Controls

<b>Controller</b>	None (Runtime Animation Controller)
<b>Avatar</b>	Teacher_BotAvatar
<b>Apply Root Motion</b>	Yes
<b>Update Mode</b>	Normal
<b>Culling Mode</b>	Cull Update Transforms

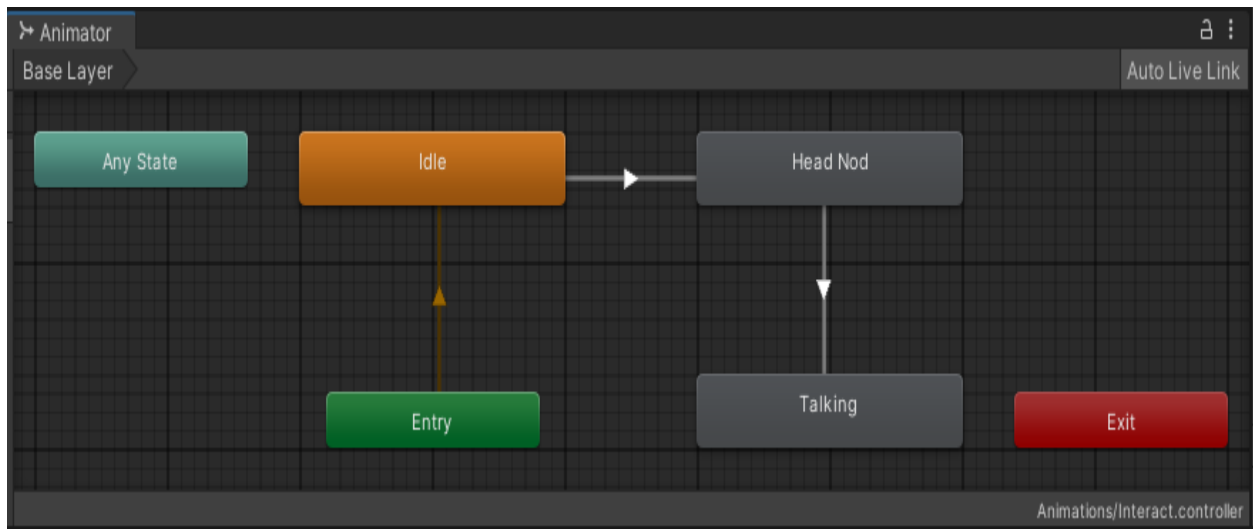
12. Now, to run the animation we need an animation controller. We create an animation controller, named ‘Interact’, inside the ‘Animations’ folder.
13. A new window pops up when an animation controller is created. This new window (basically a state machine) consists of various number of states that are essentially animations and we can move from state to another state (or we can move from one animation to another animation).
14. We copy the idle animation from ‘Animations’ folder to the animation controller window. The idle animation automatically becomes the default animation and its color changes to orange. The entry animation (green color) automatically makes a transition towards idle animation on its entry inside the animation controller window. Now, we turn on the ‘loop time’ checkbox of the idle animation so that it plays indefinitely on loop till the time it is stopped.

15. Now, we need to select the animation controller 'Interact' in the controller option present inside the animator component. This will result in successful run of the idle animation in the game view. Figure 3.11 shows the state transition diagram of idle animation.



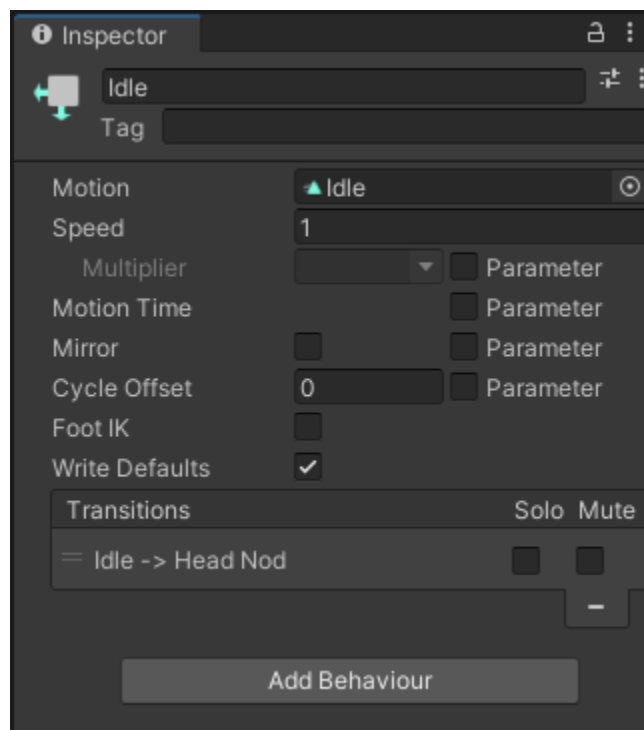
**Figure 3.11:** State Transition Diagram of Idle Animation

16. The idle animation is bereft of any interactivity, so we need to have much more animations for any kind of interactivity.
17. We need to prepare a listening animation and talking animation, so that our Teacher\_Bot character can listen to the queries of any learner and reply to them accordingly.
18. We copy the head nod (for listening) animation and talking animation from 'Animations' folder to the animation controller window. We turn on the 'loop time' checkbox for both the head nod animation and talking animation in the animation option inside the inspector window.
19. We make a transition from idle animation (orange) to head nod animation (grey) and make another transition from head nod animation (grey) to talking animation (grey).
20. Now, if we press play button it plays the animations (idle, head node and talking) and transitions from one to the other but the Teacher\_Bot character is not really interactive yet. This is just the start for making Teacher\_Bot character interactive. Figure 3.12 shows the state transition diagram of interactive animation.

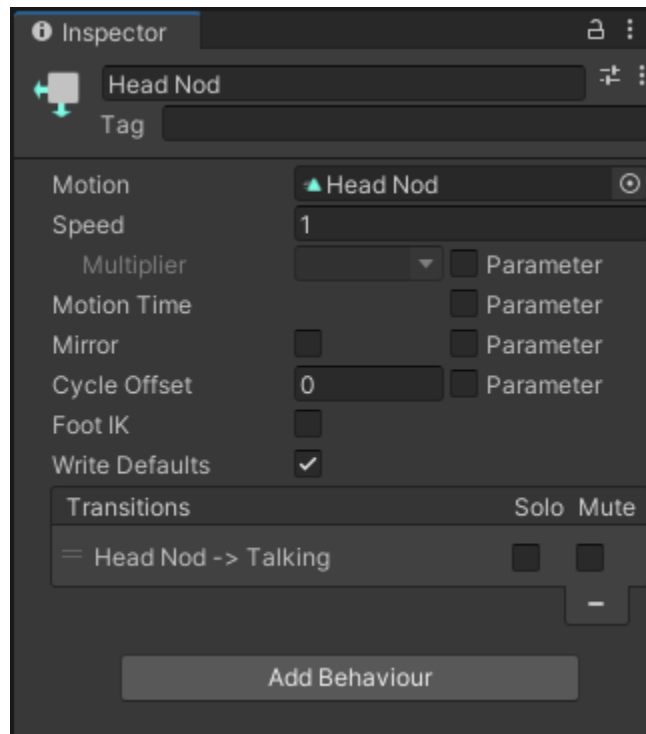


**Figure 3.12:** State Transition Diagram of Interactive Animation

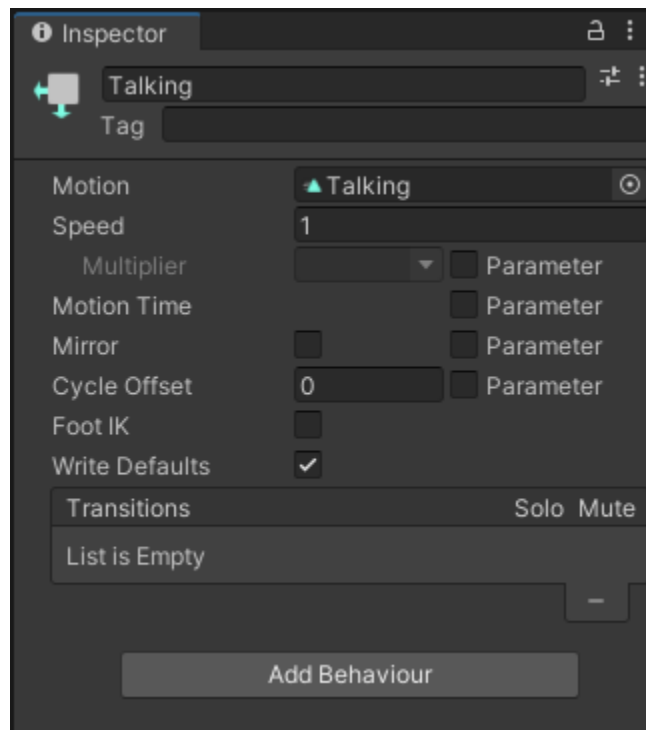
21. The various properties of idle animation, head nod animation and talking animation are visible in inspector window and displayed in Figure 3.13, 3.14 and 3.15 respectively.



**Figure 3.13:** Idle Animation Properties

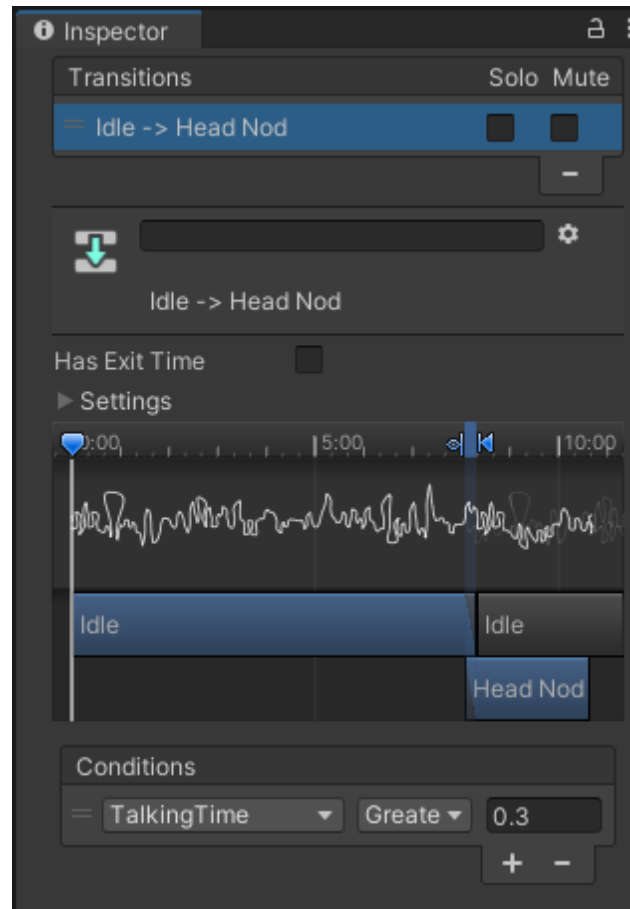


**Figure 3.14:** Head Nod Animation Properties



**Figure 3.15:** Talking Animation Properties

22. We need to set up interaction in such a way, so that the head nod (listening) and talking depends on what a learner is doing.
23. We start by choosing the idle animation from the animator controller window and select the idle -> head nod transition in the inspector window. Figure 3.16 shows the idle -> head nod transition.



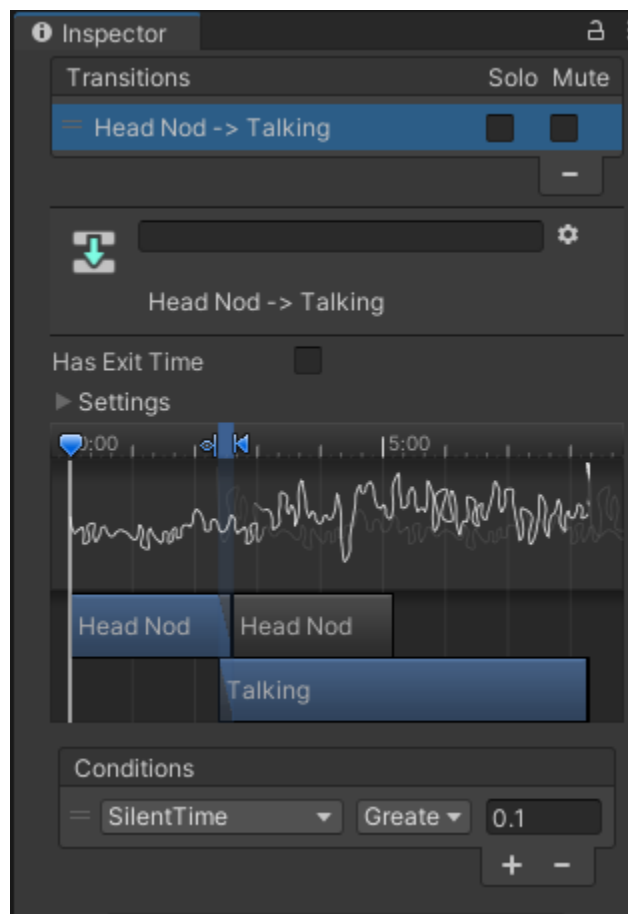
**Figure 3.16:** Idle -> Head Nod Transition

- Firstly, there is a checkbox named 'Has Exit Time' (tick marked), which plays the animation till the end, and only then will it transition. But that's not what we want, if a human starts talking, we want to interrupt straight away and the Teacher\_Bot character should listen. So, we uncheck the 'Has Exit Time' checkbox.
- Secondly, we need to put in some other condition that's going to trigger that transition. The condition relies on a parameter for it to work. Parameters are



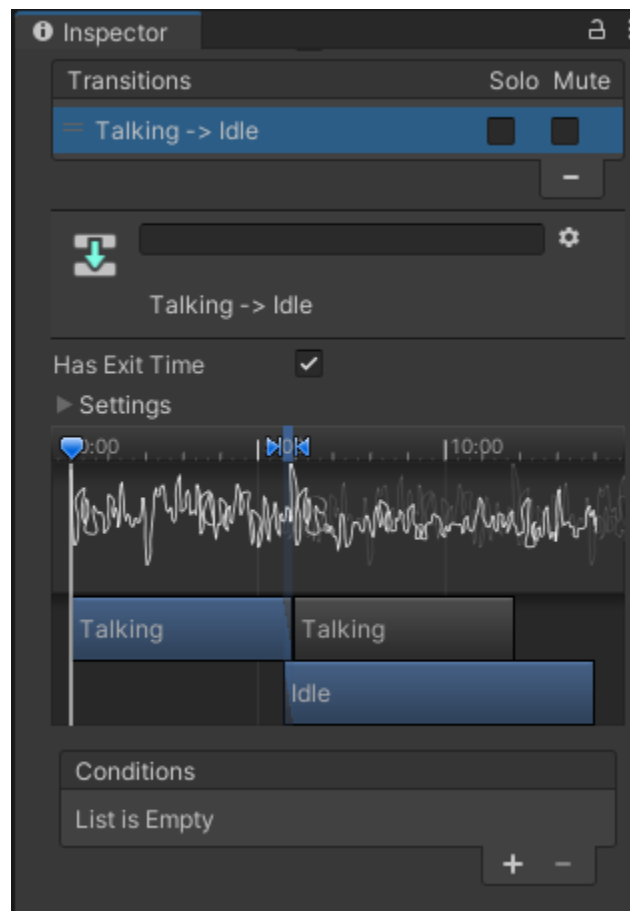
essentially variables that can control transitions and they are of various types (float, integer, boolean, and trigger). For this transition, we will use a float (basically a number) type and call it 'TalkingTime' that tells us that how long the learner has been talking for. We set the condition as 'TalkingTime > 0.3' for the idle -> head nod transition.

24. We now choose the head nod animation from the animator controller window and select the head nod -> talking transition in the inspector window. We uncheck the 'Has Exit Time' checkbox. We also need one more variable called 'SilentTime' that tells us that how long the learner has been quiet. We set the condition as 'SilentTime > 0.1' for the head nod -> talking transition. Figure 3.17 shows the head nod -> talking transition.



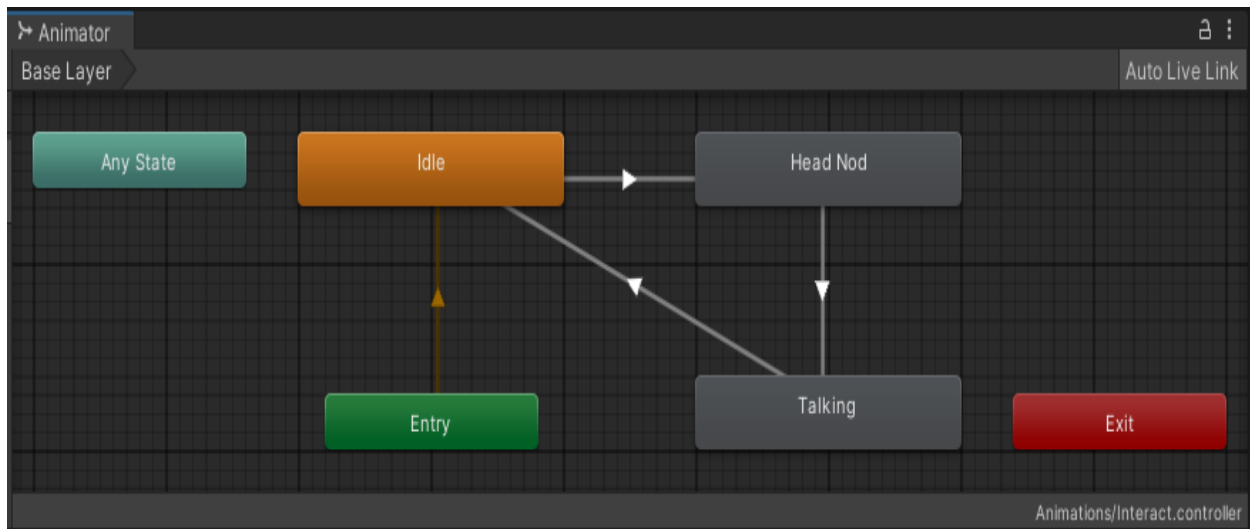
**Figure 3.17:** Head Nod -> Talking Transition

25. We now choose the head nod animation from the animator controller window and select the talking -> idle transition in the inspector window. There are no changes required as this transition happens at the end of talking i.e., the Teacher\_Bot character finishes talking, and it goes back to idle. Figure 3.17 shows the talking -> idle transition.



**Figure 3.18:** Talking -> Idle Transition

26. Now, we press the play button and change the parameters on the go to see what happens. If we change the 'TalkingTime' variable from 0 to 1 then straightaway there is a transition from idle animation to head nod animation (i.e., the Teacher\_Bot character starts listening) and if we change the 'SilentTime' variable from 0 to 1 then there is a transition from head nod animation to talking animation (i.e., the Teacher\_Bot character starts talking). Figure 3.19 shows the state transition diagram of full interactivity.



**Figure 3.19:** State Transition Diagram of Full Interactivity

27. At the moment, these parameters don't mean anything i.e., they are not going to actually change when the learner is talking. So, we need a mic input script and an audio source script to recognize the microphone input.
28. The audio source is script labeled as 'Play Audio'. This script is displayed in Figure 3.20.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

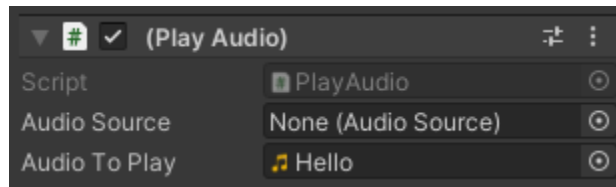
public class PlayAudio : StateMachineBehaviour {

    public AudioSource audioSource;
    public AudioClip audioToPlay;

    // OnStateEnter is called when a transition starts and the state machine starts to evaluate this state
    override public void OnStateEnter(Animator animator, AnimatorStateInfo stateInfo, int layerIndex) {
        if(audioSource == null)
        {
            audioSource = animator.GetComponent<AudioSource>();
        }
        audioSource.clip = audioToPlay;
        audioSource.Play();
    }
}
```

**Figure 3.20:** Play Audio Script

29. In the talking animation, an audio is selected to play through the 'Play Audio' script.



**Figure 3.21:** Audio Selection

30. The 'Mic Input' script is displayed in the Figures 3.22, 3.23, 3.24, 3.25, 3.26, 3.27, and 3.28.

```
using UnityEngine;
using System.Collections;

[RequireComponent(typeof(AudioSource))]
public class MicInput : MonoBehaviour {

    public float loudness;
    public string micName;

    public bool detectedSound;
    public float threshold;

    public float mean = 0;
    public float sd = 0.00000025f;
    public int n = 1;

    public float talkingTime = 0;
    public float silentTime = 0;

    Animator anim;
    public string talkingTimeParameter = "TalkingTime";
    public string silentTimeParameter = "SilentTime";
    int talkingTimeParamId;
    int silentTimeParamId;

    //public DetectSpeech detector;
    void Start()
    {
        anim = GetComponent<Animator>();
        talkingTimeParamId = Animator.StringToHash(talkingTimeParameter);
        silentTimeParamId = Animator.StringToHash(silentTimeParameter);
    }
}
```

**Figure 3.22:** Variable Declaration

```

//mic initialization
void InitMic(){
    for(int i = 0; i < Microphone.devices.Length; i++)
    {
        Debug.Log(Microphone.devices[i]);
    }
    //Debug.Log("found " + micName);
    if (micName == null || micName == "") micName = Microphone.devices[0];
    //Debug.Log (micName);
    _clipRecord = Microphone.Start(micName, true, 2, 44100);
    //Debug.Log("started mic");
    detectedSound = false;
    _isInitialized = true;
    // threshold = mean + 2 * (Mathf.Sqrt(sd));
}

void StopMicrophone()
{
    Microphone.End(micName);
}

```

Figure 3.23: Mic Initialization

```

public AudioClip _clipRecord;//= new AudioClip();
public int _sampleWindow = 128;

//get data from microphone into audioclip
float LevelMax()
{
    float levelMax = 0;
    float[] waveData = new float[_sampleWindow];
    int micPosition = Microphone.GetPosition(micName)-(_sampleWindow+1);

    if (micPosition < 0) return 0;
    _clipRecord.GetData(waveData, micPosition);
    // Getting a peak on the last 128 samples
    for (int i = 0; i < _sampleWindow; i++) {
        float wavePeak = waveData[i] * waveData[i];
        if (levelMax < wavePeak) {
            levelMax = wavePeak;
        }
    }
    return levelMax;
}

```

Figure 3.24: Audio Clip

```

void Update()
{
    if (!_isInitialized)
    {
        InitMic();
    }

    loudness = LevelMax ();

    if (Time.time > 5.0f && loudness > threshold) {
        //Debug.Log("sound");
        detectedSound = true;
        gameObject.SendMessage("Listen", null, SendMessageOptions.DontRequireReceiver);
        silentTime = 0;
        talkingTime += Time.deltaTime;
    } else
    {
        silentTime += Time.deltaTime;
        talkingTime = 0;
        if (detectedSound) {
            //print("got speech");
            StopMicrophone();

            gameObject.SendMessage("StopListening", null, SendMessageOptions.DontRequireReceiver);
        }
    }
}

```

Figure 3.25: Loudness

```

        gameObject.SendMessage("StopListening", null, SendMessageOptions.DontRequireReceiver);

        InitMic();
    } else {
        n++;
        float newMean = mean + (loudness - mean) / n;
        sd = sd + (loudness - mean) * (loudness - newMean);
        mean = newMean;
        //threshold = mean + 2 * (Mathf.Sqrt(sd / (n - 1)));
    }
    detectedSound = false;
}
anim.SetFloat(talkingTimeParamId, talkingTime);
anim.SetFloat(silentTimeParamId, silentTime);
}

bool _isInitialized;
// start mic when scene starts
void OnEnable()
{
    //InitMic();
    //_isInitialized=true;
    talkingTime = 0;
    silentTime = 0;
}

```

Figure 3.26: Mic Start

```

//stop mic when loading a new level or quit application
void OnDisable()
{
    StopMicrophone();
}

void OnDestroy()
{
    StopMicrophone();
}

```

Figure 3.27: Mic Stop

```

// make sure the mic gets started & stopped when application gets focused
void OnApplicationFocus(bool focus) {

    if (focus && enabled)
    {
        //Debug.Log("Focus");

        if(!_isInitialized){
            //Debug.Log("Init Mic");
            InitMic();
            _isInitialized=true;
        }
    }
    if (!focus)
    {
        //Debug.Log("Pause");
        StopMicrophone();
        //Debug.Log("Stop Mic");
        _isInitialized=false;
    }
}
}

```

Figure 3.28: Start and Stop Mic when Application is at Focus

### 3.3.3 Embedding the Video File in Unity 3D Virtual Platform

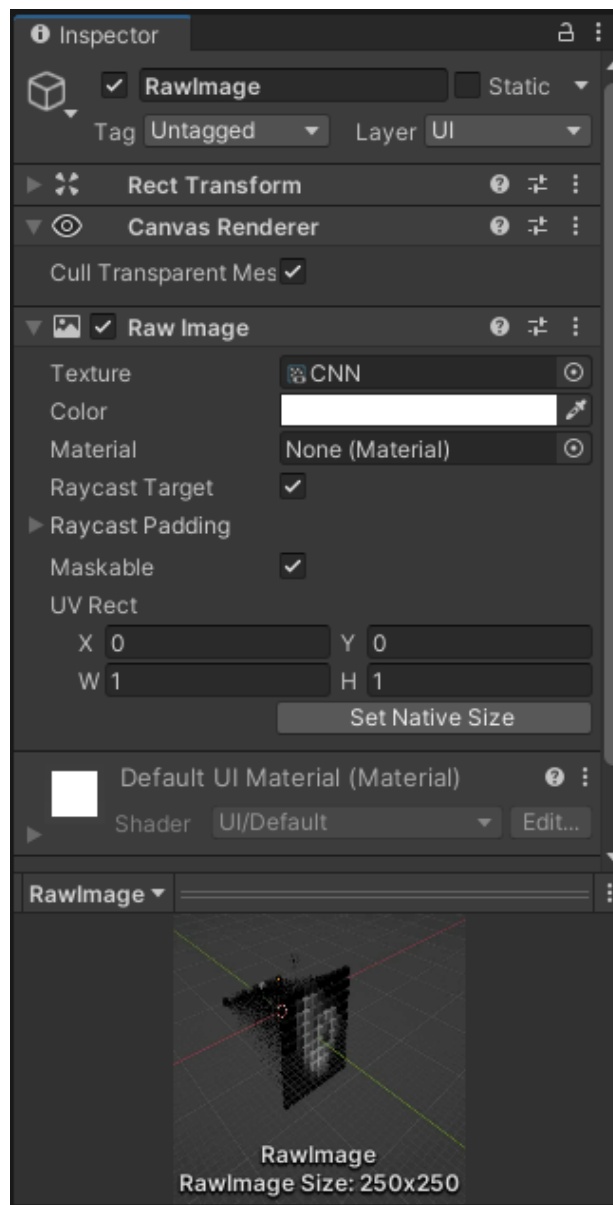
1. We create a new Render Texture inside the assets folder. We use this render texture to display the video on a UI canvas and name it as 'CNN'.
  2. We need to make the size of the render texture to be the same as the video. We select the video and in the inspector window we chose the source info option to get the information about the pixels of the video. We go back to select the render texture and add in the sizes.
  3. We now create a Video Player in the Hierarchy window. We now add our video clip in the 'Video Clip' option of the Video Player component present inside the Inspector window.
- Figure 3.29 shows the video player.



**Figure 3.29:** Video Player



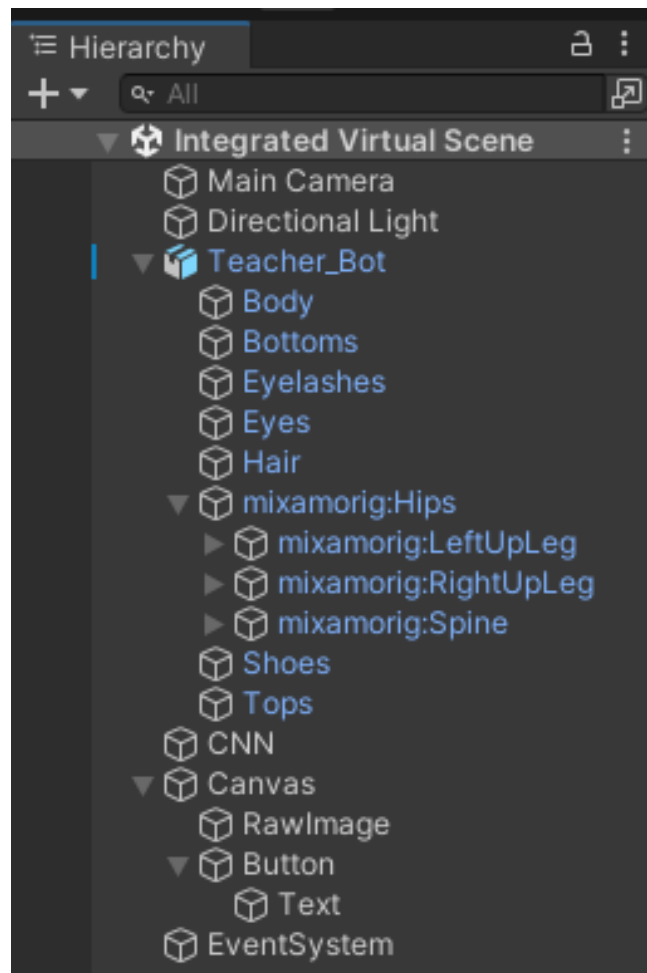
4. Next, we add render texture in the ‘Target Texture’ option of the Video Player component present inside the Inspector window.
5. Now, we create a canvas to display the video. In the Hierarchy window we create a raw image through UI.
6. Lastly, we need to add the render texture to the texture of this raw image. Figure 3.30 shows the raw image.



**Figure 3.30:** Raw Image

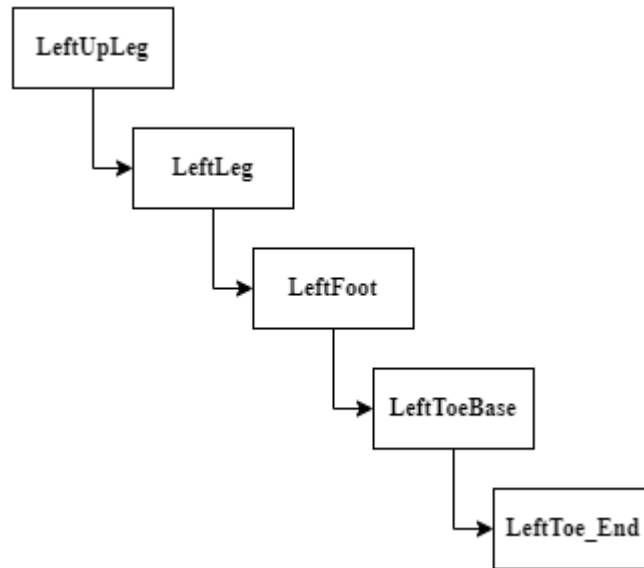
### 3.2.4 Hierarchy in the Integrated Virtual Scene

1. In Figure 3.31, it can be seen that 'Integrated Virtual Scene' is the parent and it has six children: Main Camera, Directional Light, Teacher\_Bot, CNN, Canvas, and EventSystem.
2. Main Camera, Directional Light, CNN and EventSystem have no child.
3. Teacher\_Bot consists of eight children: Body, Bottoms, Eyelashes, Eyes, Hair, Hips, Shoes and Tops. Out of these 8 children only Hips component has three children (LeftUpLeg, RightUpLeg, and Spine) while other 7 components have no child.
4. Canvas consists of two children: RawImage and Button. RawImage has no child while Button has one child.



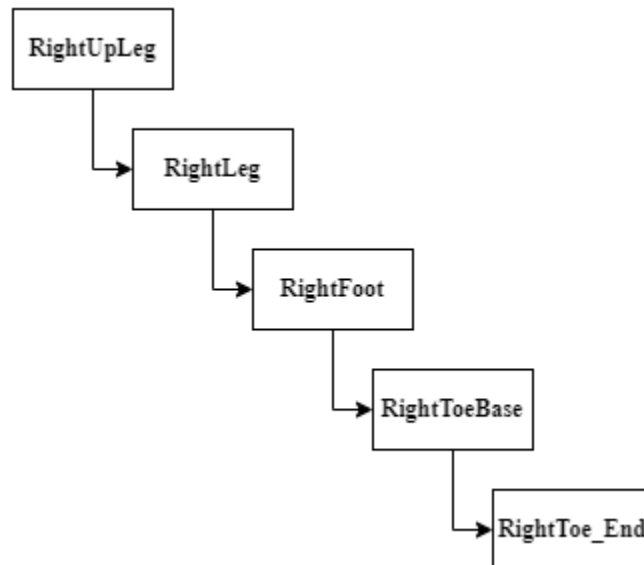
**Figure 3.31:** Hierarchy Window

5. LeftUpLeg has the following hierarchy:



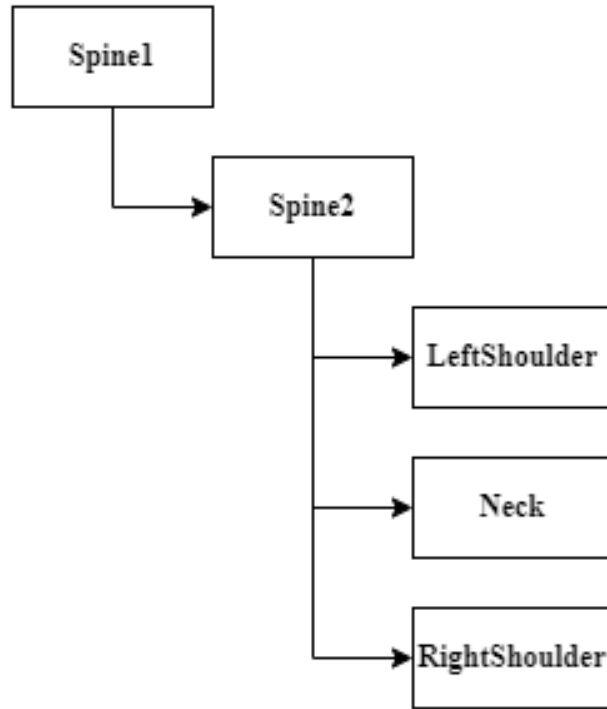
**Figure 3.32:** Hierarchy of Left Leg

6. RightUpLeg has the following hierarchy:



**Figure 3.33:** Hierarchy of Right Leg

7. Spine has the following hierarchy:



**Figure 3.34:** Hierarchy of Spine

**The transform component values of the children of the Integrated Virtual Scene are as follows:**

1. The transform component of Main Camera is detailed in table 3.6.

**Table 3.6:** Transform Component of Main Camera

	x	y	z
<b>Position</b>	0	1.5	5
<b>Rotation</b>	0	180	0
<b>Scale</b>	1	1	1

2. The transform component of Directional Light is detailed in table 3.7.

**Table 3.7:** Transform Component of Directional Light

	x	y	z
<b>Position</b>	0	3	0
<b>Rotation</b>	50	-30	0
<b>Scale</b>	1	1	1

3. The transform components of CNN, EventSystem and Teacher\_Bot is detailed in table 3.8.

**Table 3.8:** Transform Component of CNN, EventSystem and Teacher\_Bot

	x	y	z
<b>Position</b>	0	0	0
<b>Rotation</b>	0	0	0
<b>Scale</b>	1	1	1

4. The transform components of the 7 children (body, bottoms, eyelashes, eyes, hairs, shoes, and tops) of the Teacher\_Bot are similar. The transform component of last child i.e., hips is detailed in table 3.9.

**Table 3.9:** Transform Component of Hips

	x	y	z
<b>Position</b>	0.0004184151	2.090796	-0.01416565
<b>Rotation</b>	0	0	0
<b>Scale</b>	1	1	1

5. The transform components of LeftUpLeg, RightUpLeg, and Spine is detailed in tables 3.10, 3.11 and 3.12 respectively.

**Table 3.10:** Transform Components of LeftUpLeg

	x	y	z
<b>Position</b>	-0.185244	-0.1383853	0.02662015
<b>Rotation</b>	0.176	0.001	-179.727
<b>Scale</b>	1	1	1

**Table 3.11:** Transform Components of RightUpLeg

	x	y	z
<b>Position</b>	0.1844027	-0.1441169	0.001603403
<b>Rotation</b>	-1.266	-0.001	179.629
<b>Scale</b>	1	1	1

**Table 3.12:** Transform Component of Spine

	x	y	z
<b>Position</b>	0.001948024	0.1980479	-0.0388975
<b>Rotation</b>	-5.735	0.066	-0.663
<b>Scale</b>	1	1	1

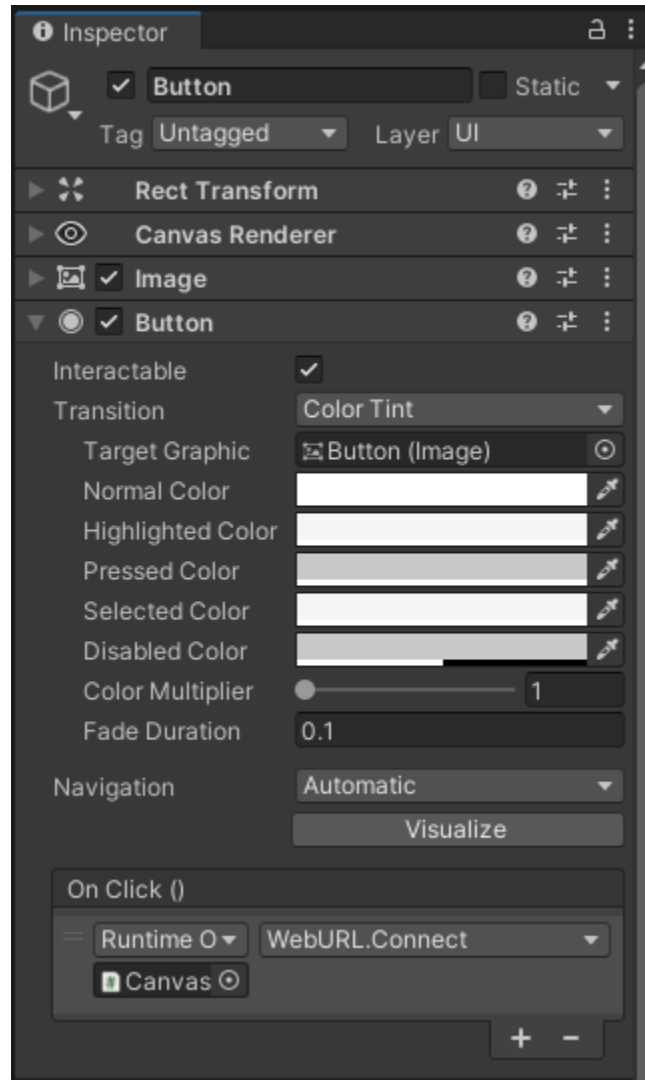
### 3.3 CONNECT WITH A FRIEND OR A TEACHER OR ANYONE

If the learner is not satisfied with the answers of Teacher\_Bot then (s)he can click the ‘Connect’ button present in the bottom-right corner of the Game View to connect with a friend or a teacher or anyone in the world to get more personalized answers for her/his questions.

How a button is embedded in the integrated virtual scene

- We create a C# script and name it ‘WebURL’.
- We now create a button through UI. This button resides inside the same Canvas which contains RawImage component.

- We now select the button and go to the 'On Click ()' option of the Button component present inside the Inspector window.
- We now drag the Canvas component inside the 'On Click ()' option of the Button component and change the function to WebURL.Connect. Figure 3.35 and 3.36 show the Connect button and Canvas respectively.



**Figure 3.35:** Connect Button

- Lastly, we drag the WebURL script to the Canvas component present inside the Hierarchy window. Figure 3.37 shows the WebURL script.

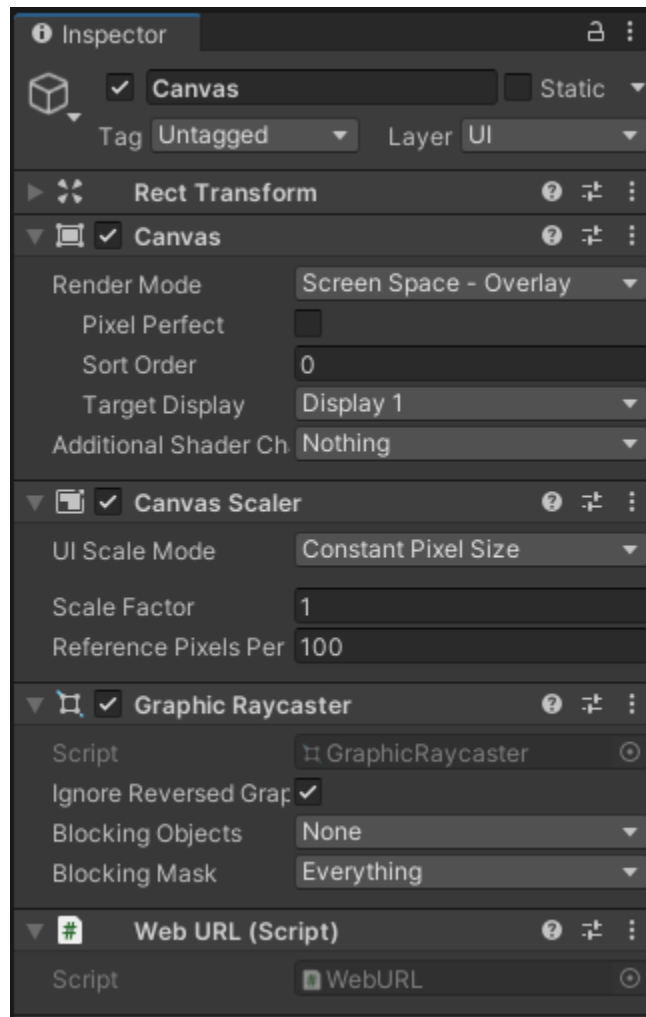


Figure 3.36: Canvas

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class WebURL : MonoBehaviour
{
    public void Connect()
    {
        Application.OpenURL("https://alcer.azurewebsites.net/");
    }
}
```

Figure 3.37: WebURL Script



- Now, when the learner clicks the connect button (s)he is redirected to a web page that contains the link to discuss with a friend, a teacher or anyone. Figures 3.39, 3.40, 3.41, and 3.42 show the web page HTML script.

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}
< style >

    .header {
        background-color: #007bff;
        padding: 15px;
        color: #fff;
        text-align: center;
    }

    body {
        font-family: Arial, sans-serif;
        background-color: #f2f2f2;
        margin: 0;
        padding: 0;
    }

    h2 {
        text-align: center;
        margin-bottom: 40px;
        color: #007bff;
        margin-top: 5%;
        font-size: -webkit-xxx-large;
    }
```

**Figure 3.38:** Web Page HTML Script

```

.admin - tiles {
display: grid;
grid - template - columns: repeat(1, 1fr);
gap: 20px;
padding: 20px;
max - width: 900px;
margin: 0 auto;
}

.admin - tile {
height: 150px;
display: flex;
justify - content: center;
align - items: center;
background - color: #007bff;
color: #fff;
border - radius: 10px;
cursor: pointer;
transition: transform 0.3s ease;
font - size: 30px;
font - weight: bold;
text - decoration: none;
box - shadow: 0px 2px 6px rgba(0, 0, 0, 0.2);
}

.admin - tile:hover {
transform: scale(1.05);
}

```

Figure 3.39: Grids

```

/* Customize colors for each tile */
.admin - tile.add - new- record {
background - color: #2ecc71;
}

.admin - tile.worksheet {
background - color: #e67e22;
}

.admin - tile.search - and - update {
background - color: #9b59b6;
}

.admin - tile.search - trademark {
background - color: #3498db;
}

.admin - tile.create - new- user {
background - color: #f1c40f;
}

.admin - tile.view - new- user {
background - color: #989669;
}
</ style >

```

Figure 3.40: Tiles

```

< h2 > Automating Learner Centric Education and Research</h2>

<div class= "admin-tiles" >
  < a class= "admin-tile worksheet" >
    < h3 > Connect With Friend</h3>
  </a>
  <a class= "admin-tile search-trademark" >
    < h3 > Connect With Teacher</h3>
  </a>
  <a class= "admin-tile add-new-record" >
    < h3 > Connect With Anyone</h3>
  </a>
</div>

```

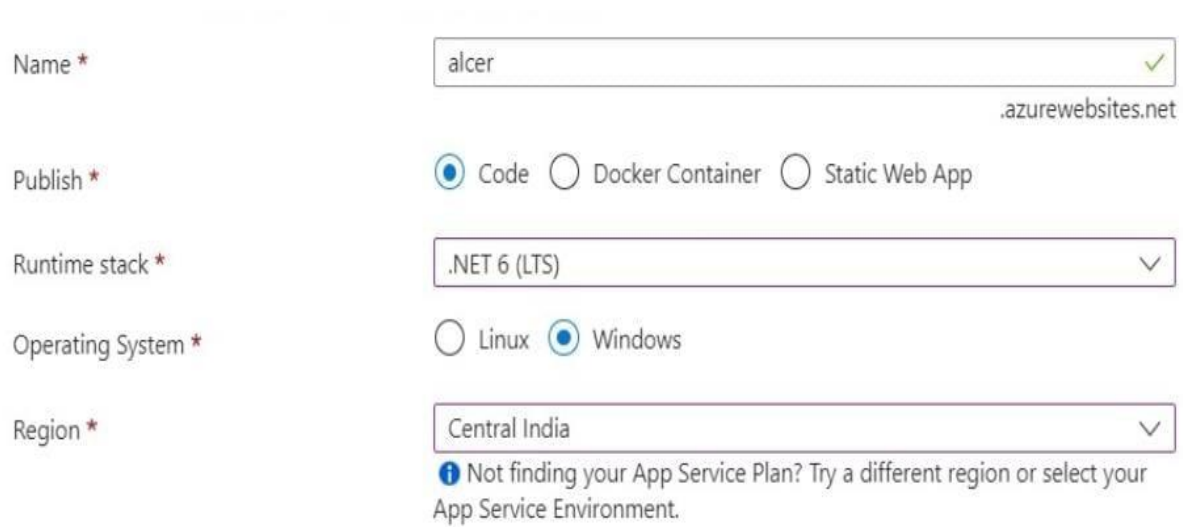
**Figure 3.41: Classes**

Figure 3.42 shows the web page.



**Figure 3.42: Web Page**

Figure 3.43 shows the web app creation.



The screenshot shows the 'Web App Creation' form in the Azure portal. It includes the following fields and options:

- Name \***: A text input field containing 'alcer' with a green checkmark icon on the right. Below the field, the text '.azurewebsites.net' is displayed.
- Publish \***: Three radio button options: 'Code' (selected), 'Docker Container', and 'Static Web App'.
- Runtime stack \***: A dropdown menu showing '.NET 6 (LTS)' with a downward arrow icon.
- Operating System \***: Two radio button options: 'Linux' and 'Windows' (selected).
- Region \***: A dropdown menu showing 'Central India' with a downward arrow icon.

Below the 'Region' dropdown, there is a blue information icon followed by the text: 'Not finding your App Service Plan? Try a different region or select your App Service Environment.'

**Figure 3.43:** Web App Creation

## **CHAPTER-4**

### **CONCLUSION AND FUTURE DIRECTIONS**

#### **4.1 CONCLUSION**

The integrated virtual scene facilitates to let any learner:

1. Watch the content like a virtual film.
2. Interact with the visual teacher bot in the voice of the teacher.
3. Seek answers for her/his questions in a sequential manner.

This is the beginning of a new education system that will benefit everybody because

- Everybody will be able to create personalized content by giving instructions through speech or touch or thought.
- Everybody will be free to learn interactively from her/his choice virtual human agent (e.g. learning special theory of relativity from virtual Einstein or enjoying any dialogue from any virtual actress) by selecting the preferred voice sample and running the text through the relevant app. (S)he can learn interactively from teacher bot also in the voice of teacher. If not satisfied with the answers by virtual human agent or teacher bot then (s)he may directly connect with the real human teacher online.
- Haptics can be integrated in a virtual platform that will enhance learning abilities.
- Since AI will slowly replace human labor, the new digital system can be created to let anyone earn for learning anything that will perpetuate creative competition, innovation and justice with health promoting entertainment.

## 4.2 FUTURE DIRECTIONS

### My plans to upgrade this initiative in near future

1. I created the content in the blender software but the integrated virtual scene will create content when it gets fully developed.
2. **Develop a user centric new UI:** Since any user will be able to do all her/his work, create personalized content and learn interactively just by giving instructions through her/his speech or touch or thought, this will create a new UI different from that of laptop and mobile that will also facilitate 3D visualization in the mid-air as well as will project the content on to any screen (like mobile, laptop, TV and smart devices etc.) or on the wall.
3. **Integrate AI/ML techniques in the virtual scene:** This user centric new UI will give the freedom to any user to get outcomes for her/his instructions like converting user inputs to codes, content generation and presentation, interactive learning etc directly that will slowly limit the use of modern day applications (Windows, Android and iOS etc.).
4. **Develop a very cheap ring as an input-output device:** Modern day computing devices (laptops and mobiles) will get slowly replaced by a wearable ring that will act as an input-output device for quantum communication and control, automation and robotics, online manufacturing etc. without the need for data processing hardware in the ring because quantum sensors will entangle the desired quantum communication with the data at the remote central server. This will be a boon for the poorest students.

## REFERENCES

- [1] Zan D., Chen B., Zhang F., Lu D., Wu B., Guan B., Wang Y., and Lou J., “Large Language Models Meet NL2Code: A Survey,” in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, Toronto, Canada, vol. 1, pp. 7443-7464, 2023.
- [2] Hossain S., Emi M., Mishu M., Zannat R. and Ohidujjaman, "Code Generator based on Voice Command for Multiple Programming Language," in *12th International Conference on Computing Communication and Networking Technologies*, Kharagpur, India, pp. 01-05, 2021.
- [3] Beltramelli T., “pix2code: Generating Code from a Graphical User Interface Screenshot,” in *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, Paris, France, pp. 01-06, 2018.
- [4] Borsos Z., Marinier R., Vincent D., Kharitonov E., Pietquin O., Sharifi M., Teboul O., Grangier D., Tagliasacchi M., and Zeghidour N., “AudioLM: a Language Modeling Approach to Audio Generation,” in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 2523-2533, 2022.
- [5] Radford A., Kim J., Xu T., Brockman G., McLeavey C., and Sutskever I., “Robust Speech Recognition via Large-Scale Weak Supervision,” in *Proceedings of the 40th International Conference on Machine Learning*, vol. 202, pp. 28492-28518, 2023.
- [6] Wang X., Qiao T., Zhu J., Hanjalic A., and Scharenborg O., “S2IGAN: Speech-to-Image Generation via Adversarial Learning,” *Interspeech*, pp. 2292-2296, 2020.
- [7] Si S., Wang J., Qu X., Cheng N., Wei W., Zhu X., and Xiao J., “Speech2Video: Cross-Modal Distillation for Speech to Video Generation,” *Interspeech*, pp. 1629-1633, 2021.
- [8] Khan S. and Tuncer B., “Gesture and speech elicitation for 3D CAD modeling in conceptual design,” *Automation in Construction*, vol. 106, 2019.
- [9] Guo M., Ainslie J., Uthus D., Ontanon S., Ni J., Sung Y., and Yang Y., “LongT5: Efficient Text-To-Text Transformer for Long Sequences,” in *Findings of the Association for Computational Linguistics: NAACL*, Seattle, United States, pp. 724–736, 2022.
- [10] Mukherjee A., Bansal S., Satpal S., and Mehta R., “Text aware Emotional Text-to-speech with BERT,” *Interspeech*, pp. 4601-4605, 2022.
- [11] Ramesh A., Dhariwal P., Nichol A., Chu C., and Chen M., “Hierarchical Text-Conditional Image Generation with CLIP Latents,” *arXiv preprint*, 2022.
- [12] Singer U., Polyak A., Hayes T., Yin X., An J., Zhang S., Hu Q., Yang H., Ashual O., Gafni O., Parikh D., Gupta S., and Taigman Y., “Make-A-Video: Text-To-Video Generation Without Text-Video Data,” in *The Eleventh International Conference on Learning Representations*, 2022.
- [13] Lin C., Gao J., Tang L., Takikawa T., Zeng X., Huang X., Kreis K., Fidler S., Liu M., and Lin T., “Magic3D: High-Resolution Text-to-3D Content Creation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 300-309, 2023.
- [14] Zhuang W., Qi J., Zhang P., Zhang B., and Tan P., “Text/Speech-Driven Full-Body Animation,” in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pp. 5956-5959, 2022.
- [15] Cudeiro D., Bolkart T., Laidlaw C., Ranjan A., and Black M., “Capture, Learning, and Synthesis of 3D Speaking Styles,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10093-10103, 2019.
- [16] Ghosh A., Cheema N., Oguz C., Theobalt C., and Slusallek P., “Synthesis of Compositional Animations from Textual Descriptions,” in *IEEE/CVF International Conference on Computer Vision*, Montreal, Canada, pp. 1376-1386, 2021.

- [17] Stocco A., “How artificial intelligence can improve web development and testing,” in *Companion Proceedings of the 3rd International Conference on the Art, Science, and Engineering of Programming*, New York, USA, Article 13, pp. 01-04, 2019.
- [18] Devlin J., Chang M., Lee K., and Toutanova K., “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in *Proceedings of NAACL-HLT*, Minneapolis, USA, pp. 4171-4186, 2019.
- [19] Brown T., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A., Agarwal S., Voss A., Krueger G., Henighan T., Child R., Ramesh A., Ziegler D., Wu J., Winter C., Hesse C., Chen M., Sigler E., Litwin M., Gray S., Chess B., Clark J., Berner C., McCandlish S., Radford A., Sutskever I., and Amodei D., “Language Models are Few-Shot Learners,” in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NY, USA, Article 159, pp. 1877–1901, 2020.
- [20] Gao Y., Sheng T., Xiang Y., Xiong Y., Wang H., and Zhang J., “Chat-REC: Towards Interactive and Explainable LLMs-Augmented Recommender System,” *arXiv preprint*, 2023.
- [21] Biswas S. and Visell Y., “Haptic Perception, Mechanics, and Material Technologies for Virtual Reality,” *Advanced Functional Materials*, vol. 31, 2021.