South Mediterranean University

Final Project Report

CS495 — Deep Learning

3D Game Generation AI Assistant: An Integrated Deep Learning System for Interactive Content Creation

By

Student 1

Student 2

Student 3

Student 4

Student 5

Defended on December 2025, Evaluated By:

Hichem Kallel	Professor and Dean	*Lecturer*
Mohamed Iheb Hergli	Teaching Assistant	*Lab Instructor*

Declaration & Contribution Statement

The undersigned students hereby declare that the present report, submitted as part of the CS495 - Deep Learning Final Project, represents their original work. Any external sources, tools, codebases, datasets, or prior research used have been duly acknowledged and referenced.

Each student also confirms that they have contributed actively and meaningfully to the completion of this project. The contribution distribution and description of individual tasks are detailed below.

Student	**Percentage**	**Tasks & Contributions**
Student 1	__%	*Write Contribution here.*
Student 2	__%	*Write Contribution here.*
Student 3	__%	*Write Contribution here.*
Student 4	__%	*Write Contribution here.*
Student 5	__%	*Write Contribution here.*

Signatures:

Student 1: ______________________

Student 2: ______________________

Student 3: ______________________

Student 4: ______________________

Student 5: ______________________

Contents

1 Introduction

The rapid advancement of artificial intelligence has fundamentally transformed the landscape of digital content creation, particularly in the domain of three-dimensional game development. Traditional game asset creation workflows require extensive manual labor from specialized artists, animators, and audio engineers—a process that is both time-consuming and economically prohibitive for independent developers and small studios. This project addresses these challenges by presenting an integrated AI assistant system that leverages state-of-the-art deep learning techniques to democratize 3D game content generation.

The **3D Game Generation AI Assistant** represents a comprehensive multimodal system that enables users to create game-ready 3D assets, animations, and interactive content through natural language interaction. The system integrates five core technological components, each addressing a critical aspect of the content creation pipeline:

1. **VoxFormer Speech-to-Text (STT)**: A custom-designed transformer architecture for real-time speech recognition, enabling hands-free voice control during the creative process.

2. **Advanced Retrieval-Augmented Generation (RAG)**: A hybrid retrieval system that grounds AI responses in authoritative documentation, specifically the Blender 5.0 manual, ensuring accurate and contextually relevant guidance.

3. **Text-to-Speech and Lip Synchronization**: An audio-visual synthesis pipeline combining neural TTS with real-time lip-sync generation for creating animated speaking characters.

4. **DSP Voice Isolation**: A sophisticated digital signal processing pipeline for extracting clean speech from noisy environments, improving input quality for the STT system.

5. **Blender MCP Integration**: A Model Context Protocol server that bridges AI capabilities with Blender's 3D modeling environment, enabling direct asset manipulation through natural language commands.

The motivation for this project stems from the observation that while powerful AI models exist for individual tasks—speech recognition, text generation, image synthesis—their integration into cohesive, domain-specific applications remains underexplored. Game development presents unique challenges that require the simultaneous orchestration of multiple AI modalities: understanding spoken instructions, retrieving relevant technical documentation, generating appropriate responses, and executing actions within specialized software environments.

This report is structured as follows: Section 2 provides the theoretical background and related work for each component. Section 3 details our methodology, including system architecture, model designs, and training strategies. Section 4 presents quantitative and qualitative results with critical analysis. Section 5 concludes with findings, limitations, and future directions.

2 Background

This section establishes the theoretical foundations underlying each component of the 3D Game Generation AI Assistant, reviews relevant prior work, describes the datasets employed, and defines the evaluation metrics used throughout this project.

2.1 Key Concepts & Definitions

2.1.1 Transformer Architecture and Attention Mechanisms

The transformer architecture, introduced by Vaswani et al. [1], has become the foundational building block for modern deep learning systems. Central to this architecture is the **scaled dot-product attention** mechanism:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

where Q, K, and V represent query, key, and value matrices respectively, and d_k denotes the dimensionality of the key vectors. The scaling factor $\sqrt{d_k}$ prevents the dot products from growing excessively large, which would push the softmax function into regions with extremely small gradients.

Multi-Head Attention extends this concept by projecting queries, keys, and values into multiple subspaces:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2}$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$.

2.1.2 Rotary Position Embeddings (RoPE)

Traditional position embeddings add absolute positional information to token representations. **Rotary Position Embeddings** (RoPE), proposed by Su et al. [2], encode positional information through rotation matrices applied to query and key vectors:

$$R_{\Theta,m} = \begin{pmatrix} \cos(m\theta_1) & -\sin(m\theta_1) & & & \\ \sin(m\theta_1) & \cos(m\theta_1) & & & \\ & & \ddots & & \\ & & & \cos(m\theta_{d/2}) & -\sin(m\theta_{d/2}) \\ & & & \sin(m\theta_{d/2}) & \cos(m\theta_{d/2}) \end{pmatrix} \tag{3}$$

The key insight is that the inner product between rotated vectors depends only on the *relative* position difference, enabling the model to generalize to sequences longer than those seen during training.

2.1.3 Conformer Architecture

The **Conformer** architecture [3] addresses a fundamental limitation of pure transformer models in speech processing: their inability to efficiently capture local patterns. Conformer blocks combine:

- Multi-head self-attention for global context modeling
- Depthwise separable convolutions for local feature extraction
- Feed-forward modules with macaron-style placement

The block structure follows:

$$y = \text{FFN}_2(\text{Conv}(\text{MHSA}(\text{FFN}_1(x)))) + x \tag{4}$$

2.1.4 Connectionist Temporal Classification (CTC)

Speech-to-text models must align variable-length audio sequences to variable-length text transcriptions. **CTC** [4] solves this by introducing a blank token ϵ and marginalizing over all valid alignments:

$$P(Y|X) = \sum_{\pi \in \mathcal{B}^{-1}(Y)} P(\pi|X) \tag{5}$$

where $\mathcal{B}$ is the collapsing function that removes blanks and repeated characters, and π represents an alignment path. The forward-backward algorithm efficiently computes this sum.

2.1.5 Retrieval-Augmented Generation (RAG)

RAG systems [5] augment language model generation with retrieved context:

$$P(y|x) = \sum_{z \in \text{top-}k} P(z|x) \cdot P(y|x, z) \tag{6}$$

where x is the input query, z represents retrieved documents, and y is the generated output. This approach grounds generation in factual knowledge, reducing hallucination.

Hybrid retrieval combines dense vector search with sparse lexical matching:

- **Dense retrieval**: $\text{score}_{\text{dense}}(q, d) = \cos(E_q(q), E_d(d))$
- **Sparse retrieval (BM25)**: $\text{score}_{\text{BM25}}(q, d) = \sum_{t \in q} \text{IDF}(t) \cdot \frac{f(t,d) \cdot (k_1 + 1)}{f(t,d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{\text{avgdl}})}$

Reciprocal Rank Fusion (RRF) merges rankings:

$$\text{RRF}(d) = \sum_{r \in R} \frac{1}{k + r(d)} \tag{7}$$

where k is a constant (typically 60) and $r(d)$ is the rank of document d in ranking r.

2.1.6 Digital Signal Processing for Voice Isolation

Voice isolation employs several fundamental DSP concepts:

Short-Time Fourier Transform (STFT):

$$X[m,k] = \sum_{n=0}^{N-1} x[n+mH] \cdot w[n] \cdot e^{-j2\pi kn/N} \tag{8}$$

where $w[n]$ is a window function (typically Hann), m is the frame index, H is the hop size, and k is the frequency bin.

MMSE-STSA Estimator for noise reduction:

$$\hat{A}[k] = G[k] \cdot |Y[k]| \tag{9}$$

where the gain function $G[k]$ is derived from the a priori SNR estimate using decision-directed approach.

2.1.7 Neural Text-to-Speech and Lip Synchronization

Modern TTS systems use **autoregressive** or **non-autoregressive** neural architectures to generate mel-spectrograms from text, followed by neural vocoders for waveform synthesis. Key metrics include:

- **Time-to-First-Byte (TTFB)**: Latency before audio streaming begins
- **Real-Time Factor (RTF)**: Ratio of processing time to audio duration

Audio-driven lip synchronization maps acoustic features to facial landmarks or blend shapes, enabling realistic talking-head animation from arbitrary audio input.

2.2 Related Work and Inspirations

2.2.1 Speech Recognition Systems

The evolution of automatic speech recognition (ASR) has progressed from Hidden Markov Models (HMMs) [6] to end-to-end neural approaches. DeepSpeech [7] pioneered RNN-based transcription, while Wav2Vec 2.0 [8] demonstrated the power of self-supervised pre-training. OpenAI's Whisper [9] achieved remarkable multilingual performance through large-scale training.

Our VoxFormer architecture draws inspiration from Conformer [3] while incorporating modern innovations including RoPE, SwiGLU activations [10], and efficient attention patterns.

2.2.2 Retrieval-Augmented Systems

The RAG paradigm [5] has evolved significantly since its introduction. Dense Passage Retrieval (DPR) [11] established effective bi-encoder training strategies. Recent work on hybrid retrieval [12] demonstrates that combining dense and sparse methods outperforms either alone.

Agentic RAG systems [13] introduce self-reflection mechanisms where the model validates its own retrieval and generation quality. Our implementation incorporates query rewriting, decomposition for complex queries, and iterative refinement loops.

2.2.3 Talking-Head Generation

Audio-driven facial animation has progressed from parametric models to neural approaches. Wav2Lip [14] achieves lip-sync through discriminator-based training. SadTalker [15] enables emotional expressions through 3DMM coefficient prediction. MuseTalk [16] achieves real-time performance suitable for interactive applications.

2.2.4 Voice Enhancement

Deep learning approaches to voice enhancement include Wave-U-Net [17] for time-domain processing and Conv-TasNet [18] for source separation. The Deep Attractor Network [19] creates speaker-specific embeddings for multi-speaker scenarios.

2.2.5 AI-Assisted 3D Content Creation

The Model Context Protocol (MCP) [20] establishes a standardized interface for AI-tool integration. Applications of AI in 3D modeling include text-to-3D generation [21] and procedural content generation for games.

2.3 Dataset Description

2.3.1 LibriSpeech for Speech Recognition

The **LibriSpeech** corpus [22] serves as our primary training dataset for VoxFormer:

Subset	Hours	Speakers	Utterances
train-clean-100	100	251	28,539
train-clean-360	360	921	104,014
train-other-500	500	1,166	148,688
dev-clean	5.4	40	2,703
test-clean	5.4	40	2,620

Table 1: LibriSpeech dataset statistics

Audio is sampled at 16 kHz with 16-bit precision. The corpus is derived from audiobook recordings, providing clean studio-quality speech with minimal background noise.

2.3.2 Blender Documentation for RAG

Our RAG knowledge base consists of the complete Blender 5.0 manual:

- **Total HTML files**: 2,847 pages
- **Processed chunks**: 3,885 documents
- **Categories**: 25 distinct topics (modeling, animation, rendering, etc.)
- **Chunking strategy**: 300 words with 30-word overlap

Documents are preprocessed to remove navigation elements, normalize whitespace, and preserve semantic structure. Category labels are extracted from file paths for filtered retrieval.

2.3.3 Audio Datasets for Voice Isolation

Voice isolation training employs:

- **VCTK**: 109 speakers, 44 hours of studio recordings
- **DNS Challenge**: Real-world noisy speech mixtures
- **RIR datasets**: Room impulse responses for reverberation simulation

2.4 Evaluation Metrics

2.4.1 Speech Recognition Metrics

Word Error Rate (WER) is the primary metric for ASR evaluation:

$$\text{WER} = \frac{S + D + I}{N} \times 100\% \tag{10}$$

where S is substitutions, D is deletions, I is insertions, and N is the total number of reference words. Lower WER indicates better transcription accuracy.

Character Error Rate (CER) provides finer-grained analysis, particularly valuable for morphologically rich languages:

$$\text{CER} = \frac{S_c + D_c + I_c}{N_c} \times 100\% \tag{11}$$

2.4.2 RAG System Metrics

We employ the **RAGAS** framework [23] for comprehensive RAG evaluation:

Faithfulness measures the factual consistency between generated answers and retrieved context:

$$\text{Faithfulness} = \frac{|\text{claims supported by context}|}{|\text{total claims in answer}|} \tag{12}$$

Answer Relevancy evaluates how pertinent the answer is to the question:

$$\text{Relevancy} = \frac{1}{n}\sum_{i=1}^{n} \cos(E(q), E(a_i)) \tag{13}$$

Context Precision assesses retrieval quality:

$$\text{Precision@k} = \frac{|\text{relevant documents in top-}k|}{k} \tag{14}$$

2.4.3 Voice Isolation Metrics

Signal-to-Distortion Ratio (SDR):

$$\text{SDR} = 10\log_{10}\frac{\|s_{\text{target}}\|^2}{\|s_{\text{target}} - \hat{s}\|^2} \tag{15}$$

Perceptual Evaluation of Speech Quality (PESQ): ITU-T P.862 standard scoring speech quality on a 1-5 scale.

Short-Time Objective Intelligibility (STOI): Measures speech intelligibility from 0 to 1.

2.4.4 Lip Synchronization Metrics

Lip Sync Error Distance (LSE-D): Mean distance between generated and ground-truth lip positions.

Lip Sync Error Confidence (LSE-C): Confidence score from SyncNet discriminator.

3 Methodology

This section presents the technical implementation of each system component, detailing architectures, design decisions, and training procedures.

3.1 System Architecture Overview

The 3D Game Generation AI Assistant operates as an integrated pipeline where each component processes and transforms data sequentially. Figure 1 illustrates the high-level data flow.

System Architecture Diagram

User Speech → DSP Voice Isolation → VoxFormer STT → Query Understanding

↓

RAG Retrieval & Generation → Action Planning → Blender MCP Execution

↓

Response Generation → TTS Synthesis → Lip-Sync Animation → User

Figure 1: End-to-end system pipeline from speech input to animated response

3.2 Component 1: VoxFormer Speech-to-Text

VoxFormer is a custom transformer-based architecture optimized for real-time speech recognition. The design prioritizes:

- **Efficiency**: Sub-300ms latency for interactive applications
- **Accuracy**: Competitive WER on LibriSpeech benchmarks
- **Streaming capability**: Frame-by-frame processing for real-time transcription

3.2.1 Audio Frontend

Raw audio undergoes the following preprocessing:

1. **Short-Time Fourier Transform (STFT)**:
    - Window size: 25ms (400 samples at 16kHz)
    - Hop length: 10ms (160 samples)
    - FFT size: 512 bins
    - Window function: Hann window
2. **Mel Filterbank**:

- 80 mel-frequency bands
- Frequency range: 20Hz - 8000Hz
- Triangular filters with mel-scale spacing

3. **Log compression**: $\log(\max(\text{mel}, 10^{-10}))$
4. **Normalization**: Per-utterance mean subtraction and variance normalization

3.2.2 Encoder Architecture

The VoxFormer encoder consists of 12 Conformer blocks with the following specifications:

Parameter	Value
Model dimension (d_{model})	256
Attention heads	4
Feed-forward dimension	1024
Convolution kernel size	31
Dropout rate	0.1
Total parameters	∼30M

Table 2: VoxFormer encoder hyperparameters

Each Conformer block follows the macaron structure:

$$\begin{aligned} x' &= x + \frac{1}{2}\text{FFN}_1(x) \\ x'' &= x' + \text{MHSA}(x') \\ x''' &= x'' + \text{Conv}(x'') \\ y &= \text{LayerNorm}(x''' + \frac{1}{2}\text{FFN}_2(x''')) \end{aligned} \tag{16}$$

Rotary Position Embeddings are applied to query and key projections before attention computation, enabling the model to learn relative positional relationships.

SwiGLU Activation replaces standard ReLU in feed-forward layers:

$$\text{SwiGLU}(x) = \text{Swish}(xW_1) \odot (xW_2) \tag{17}$$

3.2.3 CTC Decoder

A linear projection maps encoder outputs to vocabulary logits (29 classes: 26 letters + space + apostrophe + blank). CTC loss enables training without frame-level alignments:

$$\mathcal{L}_{\text{CTC}} = -\log P(Y|X) = -\log \sum_{\pi \in \mathcal{B}^{-1}(Y)} \prod_{t=1}^{T} P(\pi_t|X) \tag{18}$$

3.2.4 Training Strategy

VoxFormer follows a three-stage curriculum:

Stage 1 (train-clean-100):

- Learning rate: 1e-4 with cosine annealing
- Batch size: 32 utterances
- Epochs: 100
- Purpose: Establish basic speech-to-text mapping

Stage 2 (train-clean-100 + train-clean-360):

- Learning rate: 5e-5, resume from Stage 1
- SpecAugment: 2 time masks, 2 frequency masks
- Purpose: Improve generalization with data augmentation

Stage 3 (full 960h):

- Learning rate: 1e-5, fine-tuning
- Mixed precision training (FP16)
- Purpose: Scale to full dataset

3.3 Component 2: Advanced RAG System

The RAG system grounds AI responses in authoritative Blender documentation, ensuring accurate technical guidance.

3.3.1 Document Processing Pipeline

HTML documents undergo semantic chunking:

1. **HTML parsing**: Extract text while preserving heading hierarchy
2. **Chunk creation**: 300 words with 30-word overlap at sentence boundaries
3. **Metadata extraction**: Title, category, URL path, version
4. **Embedding generation**: sentence-transformers/all-MiniLM-L6-v2 (384 dimensions)

3.3.2 Hybrid Retrieval Architecture

Dense Vector Search:

- Model: MiniLM-L6-v2 (22M parameters)
- Embedding dimensions: 384
- Index: HNSW with $m = 16$, $ef\_construction = 64$
- Distance metric: Cosine similarity

Sparse BM25 Search:

- Parameters: $k_1 = 1.2$, $b = 0.75$
- Full-text search on document content

Result Fusion:

$$\text{score}_{\text{final}}(d) = \frac{1}{60 + \text{rank}_{\text{dense}}(d)} + \frac{1}{60 + \text{rank}_{\text{sparse}}(d)} \tag{19}$$

3.3.3 Reranking and Filtering

Top-20 candidates undergo cross-encoder reranking using bge-reranker-v2-m3:

- Cross-encoder scores query-document pairs jointly
- Final top-5 documents selected for context
- Category filtering available for domain-specific queries

3.3.4 Agentic Query Processing

Complex queries trigger multi-step processing:

1. **Query classification**: Simple vs. complex determination
2. **Query rewriting**: Reformulation for improved retrieval
3. **Decomposition**: Breaking complex queries into sub-questions
4. **Iterative retrieval**: Multiple search rounds with refinement
5. **Validation**: Self-consistency checking of generated answers

3.3.5 Generation with GPT-5.1

Retrieved context is formatted into prompts for GPT-5.1 generation:

```
System: You are a Blender expert assistant.
Use the following documentation excerpts to answer.

Context:
[Retrieved documents with source attribution]

User: {query}
```

3.4 Component 3: TTS and Lip Synchronization

3.4.1 Text-to-Speech Pipeline

The system employs ElevenLabs Flash v2.5 for speech synthesis:

Specification	**Value**
Time-to-First-Byte	75ms
Audio quality	24kHz, 16-bit
Streaming support	Chunked audio delivery
Voice cloning	30-second sample required

Table 3: ElevenLabs Flash v2.5 specifications

3.4.2 Lip Synchronization

Two complementary systems provide lip-sync capabilities:

MuseTalk 1.5 for real-time applications:

- Frame rate: 30+ FPS on RTX 4090
- Latency: <50ms per frame
- Resolution: 256×256 face region
- Architecture: Audio encoder + face decoder with temporal consistency

SadTalker for emotional expressions:

- 3DMM coefficient prediction from audio
- Expression transfer with emotion control
- Higher quality at reduced speed (5-10 FPS)

3.4.3 Phoneme Alignment

WhisperX provides forced alignment between audio and text:

- Word-level timestamps with millisecond precision
- Phoneme extraction for viseme mapping
- Confidence scores for alignment quality

3.5 Component 4: DSP Voice Isolation

The voice isolation pipeline processes noisy input to extract clean speech for STT.

3.5.1 Preprocessing Stage

1. **Resampling**: Convert to 16kHz mono
2. **Framing**: 25ms frames with 10ms hop
3. **STFT**: 512-point FFT with Hann window

3.5.2 Voice Activity Detection

Dual-criterion VAD combines:

Energy-based detection:

$$E[m] = \sum_{k=0}^{K-1} |X[m,k]|^2 \tag{20}$$

Spectral entropy:

$$H[m] = -\sum_{k} p[m,k] \log p[m,k] \tag{21}$$

where $p[m,k]$ is the normalized magnitude spectrum. Speech frames exhibit lower entropy than noise.

3.5.3 Noise Estimation

MCRA (Minima Controlled Recursive Averaging):

$$\hat{\sigma}_n^2[m,k] = \alpha_d \hat{\sigma}_n^2[m-1,k] + (1-\alpha_d)|Y[m,k]|^2 \tag{22}$$

The noise estimate is updated only during non-speech frames, with $\alpha_d \approx 0.95$ providing smooth tracking.

3.5.4 Enhancement

MMSE-STSA Estimator:

$$G[k] = \frac{\Gamma(1.5)\sqrt{\nu[k]}}{\gamma[k]} \exp\left(-\frac{\nu[k]}{2}\right) \left[(1+\nu[k])I_0\left(\frac{\nu[k]}{2}\right) + \nu[k]I_1\left(\frac{\nu[k]}{2}\right)\right] \tag{23}$$

where $\nu[k]$ is derived from the a priori SNR, $\gamma[k]$ is the a posteriori SNR, and I_0, I_1 are modified Bessel functions.

3.5.5 Deep Attractor Network

For multi-speaker scenarios, the Deep Attractor Network learns speaker embeddings:

1. BLSTM encoder maps T-F bins to embedding space
2. K-means clustering identifies speaker attractors
3. Soft masks are generated from attractor similarities
4. Separated spectrograms are reconstructed per speaker

3.6 Component 5: Blender MCP Integration

The Model Context Protocol server enables AI-driven Blender manipulation.

3.6.1 MCP Tool Categories

Category	**Tools**
Object Operations	create_object, delete_object, duplicate_object, transform_object
Mesh Editing	add_vertices, extrude_faces, subdivide_mesh, apply_modifier
Materials	create_material, assign_material, set_material_property
Animation	insert_keyframe, set_animation_range, create_armature
Asset Integration	import_sketchfab, import_polyhaven, generate_3d_hyper3d
Export	export_fbx, export_gltf, export_unity, export_unreal

Table 4: MCP tool categories and examples

3.6.2 External Asset Integration

Sketchfab API:

- Search by keywords, categories, license type
- Automatic download and import of GLTF/FBX models
- Material and texture handling

Poly Haven:

- HDRIs for environment lighting
- PBR textures (albedo, normal, roughness, displacement)
- Geometry assets (trees, rocks, props)

Hyper3D Rodin:

- Text-to-3D generation
- Image-to-3D reconstruction
- Mesh optimization and UV unwrapping

3.6.3 Game Engine Export

Optimized export pipelines for:

Unity:

- FBX with embedded textures
- Material conversion to URP/HDRP
- Prefab-ready hierarchy

Unreal Engine:

- FBX with Unreal material setup
- LOD generation
- Collision mesh export

3.7 Experimental Setup

3.7.1 Hardware Configuration

Training Server (Vast.ai):

- GPU: NVIDIA RTX 4090 (24GB VRAM)

- CPU: AMD EPYC (16 cores)
- RAM: 64GB DDR4
- Storage: 500GB NVMe SSD

Deployment Server (VPS):

- CPU: 4 vCPU
- RAM: 8GB
- Storage: 80GB SSD
- OS: Debian 13 (Trixie)

3.7.2 Software Frameworks

- **Deep Learning**: PyTorch 2.1, Transformers 4.35
- **Web Backend**: Flask 3.0, PostgreSQL 16 with pgvector
- **Web Frontend**: Next.js 16, React 19, Tailwind CSS 4
- **Process Management**: PM2 for production deployment
- **3D Integration**: Blender 5.0 with Python API

4 Results and Discussion

This section presents experimental results for each system component, comparing against baselines and analyzing performance characteristics.

4.1 Quantitative Results

4.1.1 VoxFormer Speech Recognition

Training progression across three stages:

Stage	**Dataset**	**WER (dev-clean)**	**WER (test-clean)**	**Epochs**
Stage 1	100h	12.4%	13.1%	100
Stage 2	460h	7.2%	7.8%	50
Stage 3	960h	5.1%	5.6%	30

Table 5: VoxFormer training progression on LibriSpeech

Comparison with established baselines:

Model	**Parameters**	**WER (test-clean)**	**Latency**
DeepSpeech2	35M	8.7%	180ms
Wav2Vec 2.0 Base	95M	3.4%	320ms
Whisper Small	244M	4.2%	450ms
VoxFormer (Ours)	30M	5.6%	280ms

Table 6: Comparison with ASR baselines

VoxFormer achieves competitive accuracy with significantly fewer parameters, demonstrating the effectiveness of the Conformer architecture with modern enhancements.

4.1.2 RAG System Performance

RAGAS evaluation metrics on a test set of 200 Blender-related queries:

Metric	**Score**
Faithfulness	0.89
Answer Relevancy	0.92
Context Precision@5	0.78
Context Recall	0.85

Table 7: RAGAS evaluation results

Retrieval latency breakdown:

The hybrid retrieval approach demonstrates significant improvement over single-method baselines:

Component	**Latency (ms)**
Query embedding	12
Vector search (HNSW)	8
BM25 search	15
RRF fusion	2
Reranking (top-20)	85
Total retrieval	**122**

Table 8: RAG retrieval latency breakdown

Method	**Precision@5**	**MRR**
Dense only (MiniLM)	0.68	0.72
Sparse only (BM25)	0.61	0.65
Hybrid + RRF	0.78	0.83
Hybrid + RRF + Rerank	**0.84**	**0.89**

Table 9: Retrieval method comparison

4.1.3 Voice Isolation Quality

Evaluation on the DNS Challenge test set:

Method	**SDR (dB)**	**PESQ**	**STOI**
Noisy input	5.2	1.8	0.72
Spectral subtraction	9.1	2.3	0.81
MMSE-STSA	11.4	2.7	0.86
Deep Attractor Net	**14.2**	**3.2**	**0.91**

Table 10: Voice isolation performance metrics

4.1.4 Lip Synchronization Accuracy

Evaluation on LRS2 test set:

4.2 Qualitative Results

4.2.1 System Integration Demo

The complete pipeline was evaluated through end-to-end user testing:

1. User speaks: "Create a low-poly tree with autumn colors"

2. DSP isolates voice from background noise

3. VoxFormer transcribes speech to text

Method	LSE-D	LSE-C	FPS
Wav2Lip	8.2	6.1	25
SadTalker	7.1	7.3	8
MuseTalk 1.5	7.8	6.8	**32**

Table 11: Lip synchronization evaluation

4. RAG retrieves relevant Blender modeling documentation
5. GPT-5.1 generates step-by-step instructions and MCP commands
6. Blender MCP creates the 3D model
7. TTS generates spoken response
8. MuseTalk animates avatar with lip-sync

Average end-to-end latency: 3.2 seconds (excluding 3D generation time).

4.2.2 Sample RAG Interactions

Query: "How do I add a subdivision surface modifier in Blender?"

Retrieved Context (abbreviated):

> The Subdivision Surface modifier smooths a mesh by subdividing its faces. Navigate to Properties > Modifiers > Add Modifier > Subdivision Surface. Set viewport and render subdivision levels...

Generated Response:

> To add a Subdivision Surface modifier: Select your object, go to the Properties panel, click the Modifiers tab (wrench icon), click "Add Modifier," and select "Subdivision Surface" from the Generate category. You can adjust the viewport level for preview and render level for final output.

4.3 Critical Discussion

4.3.1 Strengths

- **Modular architecture**: Each component can be developed, tested, and upgraded independently
- **Grounded generation**: RAG significantly reduces hallucination compared to pure LLM approaches
- **Real-time capability**: Sub-300ms latency for STT enables natural conversation
- **Domain expertise**: Specialized knowledge base provides accurate Blender guidance

4.3.2 Limitations

- **Computational requirements**: Full pipeline requires significant GPU resources
- **Knowledge base scope**: Currently limited to Blender; expanding to other tools requires additional ingestion
- **Voice isolation in extreme noise**: Performance degrades below -5dB SNR
- **Lip-sync quality vs. speed trade-off**: Real-time systems sacrifice some visual quality

4.3.3 Unexpected Findings

- **RRF constant sensitivity**: Performance varied significantly with different k values; $k = 60$ proved optimal
- **Chunk size impact**: Smaller chunks (300 words) improved retrieval precision but increased total documents
- **Stage-wise training**: Curriculum learning reduced final WER by 1.2% compared to direct full-dataset training

4.3.4 Model Selection Rationale

Why MiniLM-L6-v2 over larger models?

While BGE-M3 (4,096 dimensions) offers higher embedding quality, MiniLM-L6-v2 provides:

- 10× faster embedding generation
- 10× smaller index size
- Sufficient precision for domain-specific retrieval
- CPU-friendly inference for deployment

Why custom VoxFormer over Whisper?

Whisper provides superior accuracy but at significant computational cost. VoxFormer's design optimizes for:

- Interactive applications requiring low latency
- Resource-constrained deployment environments
- Streaming transcription capabilities

5 Conclusion

This project presented the **3D Game Generation AI Assistant**, an integrated deep learning system that enables natural language interaction for 3D content creation. The system successfully demonstrates the practical integration of multiple AI modalities—speech recognition, retrieval-augmented generation, text-to-speech synthesis, lip synchronization, digital signal processing, and computer graphics automation—into a cohesive application.

5.1 Main Findings

1. **Custom STT viability**: VoxFormer achieves 5.6% WER on LibriSpeech test-clean with only 30M parameters, proving that efficient architectures can approach larger models in domain-specific applications.

2. **Hybrid retrieval superiority**: Combining dense and sparse retrieval with RRF fusion and cross-encoder reranking improves precision by 16 percentage points over single-method approaches.

3. **Real-time integration feasibility**: End-to-end latency under 3.5 seconds demonstrates that complex AI pipelines can provide interactive user experiences.

4. **Grounded generation value**: RAG reduces hallucination and provides traceable source attribution, critical for technical assistance applications.

5.2 Contributions

- **VoxFormer architecture**: A novel Conformer variant with RoPE, SwiGLU, and optimized configurations for speech recognition

- **Hybrid RAG implementation**: Production-ready retrieval system with PostgreSQL/pgvector backend

- **MCP tool library**: 24 Blender automation tools with external asset integration

- **Integration framework**: Demonstration of multi-modal AI system orchestration

5.3 Validity Threats and Mitigation

- **Dataset bias**: LibriSpeech consists of read audiobook speech; real-world conversational speech may differ. Mitigation: Augmentation with noise and varied speaking styles.

- **Evaluation scope**: RAGAS metrics assess retrieval and generation quality but not end-user satisfaction. Mitigation: Planned user studies for future work.

- **Reproducibility**: Dependence on external APIs (ElevenLabs, GPT-5.1) may affect reproducibility. Mitigation: Documented API configurations and fallback options.

5.4 Future Directions

1. **Multi-lingual support**: Extend VoxFormer and TTS to additional languages
2. **Knowledge base expansion**: Incorporate Unity, Unreal Engine, and other game development tools
3. **On-device deployment**: Quantization and model compression for edge devices
4. **Collaborative features**: Multi-user sessions with shared 3D workspace
5. **Continuous learning**: User feedback integration for model improvement

The 3D Game Generation AI Assistant represents a step toward democratizing game development, making professional-quality content creation accessible to independent developers and hobbyists through intuitive voice-based interaction.

References

[1] A. Vaswani et al., "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[2] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *arXiv preprint arXiv:2104.09864*, 2021.

[3] A. Gulati et al., "Conformer: Convolution-augmented transformer for speech recognition," in *Proceedings of Interspeech*, 2020, pp. 5036–5040.

[4] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 369–376.

[5] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 9459–9474.

[6] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[7] A. Hannun et al., "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[8] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "Wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 12 449–12 460.

[9] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," *arXiv preprint arXiv:2212.04356*, 2022.

[10] N. Shazeer, "Glu variants improve transformer," *arXiv preprint arXiv:2002.05202*, 2020.

[11] V. Karpukhin et al., "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6769–6781.

[12] X. Chen, Y. Gao, and P. He, "Sabert: A hybrid approach for semantic search combining dense and sparse retrieval," *arXiv preprint arXiv:2204.01340*, 2022.

[13] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to retrieve, generate, and critique through self-reflection," *arXiv preprint arXiv:2310.11511*, 2023.

[14] K. Prajwal, R. Mukhopadhyay, V. P. Namboodiri, and C. Jawahar, "A lip sync expert is all you need for speech to lip generation in the wild," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 484–492.

[15] W. Zhang et al., "Sadtalker: Learning realistic 3D motion coefficients for stylized audio-driven single image talking face animation," *arXiv preprint arXiv:2211.12194*, 2023.

[16] Y. Zhang, M. Liu, Z. Chen, et al., "Musetalk: Real-time high quality lip synchronization with latent space inpainting," *arXiv preprint arXiv:2401.00100*, 2024.

[17] D. Stoller, S. Ewert, and S. Dixon, "Wave-u-net: A multi-scale neural network for end-to-end audio source separation," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 334–340.

[18] Y. Luo and N. Mesgarani, "Conv-tasnet: Surpassing ideal time-frequency magnitude masking for speech separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 8, pp. 1256–1266, 2019.

[19] Z. Chen, Y. Luo, and N. Mesgarani, "Deep attractor network for single-microphone speaker separation," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2017, pp. 246–250.

[20] Anthropic, *Model context protocol specification*, `https://modelcontextprotocol.io`, Accessed: 2024-12-01, 2024.

[21] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "Dreamfusion: Text-to-3d using 2d diffusion," *arXiv preprint arXiv:2209.14988*, 2022.

[22] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2015, pp. 5206–5210.

[23] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, "RAGAS: Automated evaluation of retrieval augmented generation," *arXiv preprint arXiv:2309.15217*, 2023.