arXiv:2403.12667v2 [cs.MM] 20 Mar 2024

# ICE: Interactive 3D Game Character Editing via Dialogue

Haoqian Wu[1], Yunjie Wu[1], Zhipeng Hu[1,2], Lincheng Li[⋆][1], Weijie Chen[1],
Rui Zhao[3], Changjie Fan[1], and Xin Yu[4]

[1] Fuxi AI Lab, Netease, Zhejiang, China
`{wuhaoqian, wuyunjie, zphu, lilincheng, chenweijie05, fanchangjie}@corp.netease.com`
[2] Zhejiang University, Zhejiang, China
[3] National University of Singapore, Singapore `rui.zhao@u.nus.edu`
[4] University of Queensland, QLD, Australia `xin.yu@uq.edu.au`

**Abstract.** Text-driven in-game 3D character auto-customization systems eliminate the complicated process of manipulating intricate character control parameters. However, current methods are limited by their single-round generation, incapable of further editing and fine-grained modification. In this paper, we propose an Interactive Character Editing framework (ICE) to achieve a multi-round dialogue-based refinement process. In a nutshell, our ICE offers a more user-friendly way to enable players to convey creative ideas iteratively while ensuring that created characters align with the expectations of players. Specifically, we propose an Instruction Parsing Module (IPM) that utilizes large language models (LLMs) to parse multi-round dialogues into clear editing instruction prompts in each round. To reliably and swiftly modify character control parameters at a fine-grained level, we propose a Semantic-guided Low-dimension Parameter Solver (SLPS) that edits character control parameters according to prompts in a zero-shot manner. Our SLPS first localizes the character control parameters related to the fine-grained modification, and then optimizes the corresponding parameters in a low-dimension space to avoid unrealistic results. Extensive experimental results demonstrate the effectiveness of our proposed ICE for in-game character creation and the superior editing performance of ICE. Project page: `https://iceedit.github.io/`.

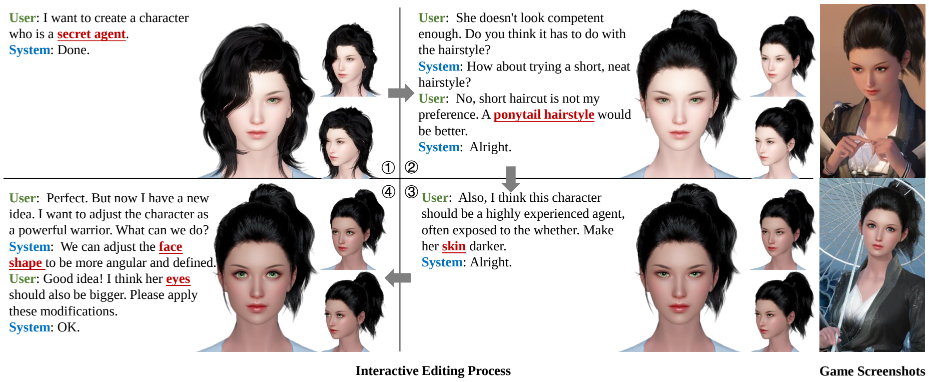
**Keywords:** 3D game character customization · 3D content generation · large language models · deep learning

## 1 Introduction

Creating a customized in-game character that mirrors the specific vision of the player is an engaging component of modern role-playing video games, AR/VR,

---

[⋆] Corresponding author



① **User**: I want to create a character who is a **secret agent**.
**System**: Done.

② **User**: She doesn't look competent enough. Do you think it has to do with the hairstyle?
**System**: How about trying a short, neat hairstyle?
**User**: No, short haircut is not my preference. A **ponytail hairstyle** would be better.
**System**: Alright.

③ **User**: Also, I think this character should be a highly experienced agent, often exposed to the whether. Make her **skin** darker.
**System**: Alright.

④ **User**: Perfect. But now I have a new idea. I want to adjust the character as a powerful warrior. What can we do?
**System**: We can adjust the **face shape** to be more angular and defined.
**User**: Good idea! I think her **eyes** should also be bigger. Please apply these modifications.
**System**: OK.

**Fig. 1:** An example of the process of our ICE is shown on the left: A character, expected as a “secret agent”, is created initially and then sequentially refined in a fine-grained and interactive manner, according to the editing instructions provided by users and suggestions of the system. Screenshots of the final generated character driven with various animations and expressions in the game are shown on the right.

and metaverses. These characters, controlled by intricate parameters ranging from facial bone position to lip colors, offer players an immersive gaming experience. Although hundreds of adjustable parameters offer a high degree of customization, manually adjusting them is very time-consuming and labor-intensive.

Recently, in-game character auto-creation systems have been developed to eliminate the need for players to operate hundreds of character control parameters. Some methods [2, 21–24, 28] automatically create 3D characters based on a reference face image, while other works [7, 31, 32] allow users to generate specific avatars based on text descriptions. However, they are single-round approaches, incapable of further editing and fine-grained modifications, and thus restrict players from incrementally articulating their ideas and precisely customizing their characters. Additionally, such approaches can lead to characters that do not align with intricate or multifaceted descriptions, as illustrated in Fig. 4b. Besides, most of them may not be applied in the game systems to help players customize characters because they may generate unrealistic results and take a long time.

To address these problems, we propose an Interactive Character Editing framework (ICE) to enable players to edit 3D game characters in a fine-grained and iterative fashion through a multi-round dialogue. In contrast to prior single-round approaches, our interactive approach has the following advantages: (1) Benefiting from our fine-grained control of character customization, ICE enables progressive editing and allows players/game asset creators to refine their ideas even after character creation. (2) Thanks to the advanced knowledge embedded in large language models (LLMs), ICE can provide detailed editing instructions even when users only give vague, high-level ideas. (3) Since our framework is designed for game systems, it directly optimizes the parameters of in-game characters (*e.g.*, retro-styled characters in our work) rather than conventional 3DMM models. As a result, our generated characters can be seamlessly incorporated into existing game systems with minimal effort. An example of our interactive editing process is depicted in Fig. 1.

Our framework contains two core components, an Instruction Parsing Module (IPM) and a Semantic-guided Latent Parameter Solver (SLPS). The proposed IPM is designed to parse interactive dialogues and then output accurate text prompts for in-game character generation. To this end, we introduce LLMs to handle multi-round dialogues and generate clear editing instruction prompts in each round. To support players to continuously refine some attributes, we design a character attribute memory bank that tracks editing states of mentioned attributes to prevent LLMs from the forgetting issue. Besides, the IPM interacts with players in dialogue and can provide suggestions to inspire players.

Our SLPS is introduced to generate and modify the parameters of a character according to the parsed editing instruction provided by IPM. To be specific, SLPS utilizes a network to localize modification-related parameters, and then optimizes them in a differentiable manner until the rendered character aligns with the parsed instruction in a pre-trained CLIP embedding space. The differentiable process relies on a neural rendering network that simulates character rendering from parameters by the game engine, facilitating cost-effective integration into various existing games. To eliminate unrealistic results, we propose to optimize character control parameters within a projected low-dimension space ensuring outcomes reflect character distributions. Our comprehensive experimental results, accompanied by ablation studies, reinforce the superiority of ICE in terms of accuracy, robustness, and user experience over single-round methods.

Our contributions are summarized as follows:

- We propose an interactive character editing framework, ICE, that enables users to interactively and fine-grained modify their 3D game characters through a multi-round dialogue. To the best of our knowledge, we are the first to study interactive 3D game character editing.
- The proposed SLPS allows for fine-grained control over character editing while considering practical application in games. It provides reliable results within an acceptable response time and is compatible with existing game systems at a low cost.
- The proposed interactive character editing framework promotes a user-friendly character creation way and facilitates fine-grained customization of 3D game characters.

## 2 Related Work

### 2.1 Game Character Auto-Creation

The auto-creation of 3D characters has recently emerged as a pivotal research topic. Various methods [2, 21–24, 28] are introduced for deriving character facial parameters from input images. Recent approaches delve into text-driven character generation, leveraging the capabilities of pretrained multimodal representation and generation models, e.g., CLIP [18] and Stable Diffusion [20]. AvatarCLIP [7] employs NeuS [26] for implicit avatar representation, incorporating a CLIP-guide loss to achieve avatar generation. Subsequent works [3,6,11] further capitalize on the SDS loss [17] to optimize the implicit representation. Rodin [27] uses diffusion models to map the shared CLIP embedding to implicitly represented avatars. However, implicit representation of characters falls short in quality and lacks compatibility with conventional graphics workflows. [12,31] utilize differential parameterized human models paired with SDS loss to produce animatable avatars. Although Dreamface [31] provides a multi-round dialogue to take user input in the online demo, the 3D generation is single-round without fine-grained editing. Applicable to any game, T2P [32] first trains a network to mimic the rendering pipeline of game engines and then search parameters to minimize a CLIP-guide loss. Nevertheless, long run time and unstable quality of these methods hinder user-friendly character customization in games. In contrast, our approach is swift in response and robust.

A crucial distinction to note is that existing methods predominantly operate in a static, single-round fashion, necessitating players to depend on exhaustive instructions or photos in a single step. Contrarily, our proposed method introduces a character creation pathway that is dynamic, supporting interactive editing.

### 2.2 Multimodal Content Editing

In the field of image processing, some methods [14,15,19,30] have explored content editing based on text instructions. [4,5,8,9,33] delve further into interactive image editing. However, these methods often struggle to accurately define the editing area and attributes, which may lead to unnecessary modifications or failures to complete modifications. Moreover, these methods are specific to the image field and cannot be applied to game characters, which are parameterized, 3D, and must adhere to specific game art styles.

Character editing is relatively less explored. Rodin [27] assumes colinearity between the CLIP embeddings of images and texts, utilizing the delta text embedding to derive the desired manipulated output. TADA [12] approaches avatar editing by adjusting the associated text prompts directly. HeadSculpt [6] introduces identity-aware editing score distillation that utilizes both the editing instructions and the initial text prompt to preserve the identity of the character. However, these methods also often suffer from inaccurate edits and superfluous modifications. In addition, they are single-step editing methods. Our method, in contrast, allows for continuous and fine-grained adjustments, providing refined editing control.

## 3 Methodology

### 3.1 Overview

As previously emphasized, our proposed interactive character editing system, ICE, differs from existing single-round creation methods by enabling users to edit

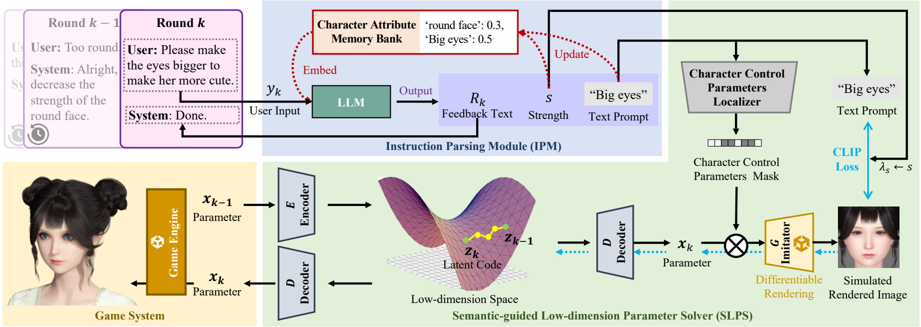


**Fig. 2: Inference Framework.** User input is first parsed by IPM as actionable editing text prompt, editing strength, and feedback text. Sequentially, SLPS localizes the character control parameters related to the specified fine-grained modification, and then optimizes them in a low-dimension space in a differentiable manner. Finally, the parameters are applied to the game engine to render the edited character.

character control parameters interactively with multi-round dialogue. Given a sequence of user-provided text instructions $Y = \{y_0, y_1, \ldots, y_K\}$, comprising an initial character text description $y_0$ and subsequent edit instructions, our system $\mathcal{M}$ can sequentially edit character control parameters and provide feedback text $R_k$ in response to user input instructions $y_k$, denoted as

$$(\hat{\boldsymbol{x}}_k, R_k) = \mathcal{M}(\hat{\boldsymbol{x}}_{k-1}, y_k). \tag{1}$$

Here, $\hat{\boldsymbol{x}}_k \in \mathbb{R}^N$ denotes a set of parameters that customizes the game character, encompassing elements like bone positions, makeup types, and so forth. The editing system then visualizes the character through the game engine based on the generated parameters. Initially, character control parameters $\hat{\boldsymbol{x}}_0$ are generated directly from the input text $y_0$, denoted as $(\hat{\boldsymbol{x}}_0, R_0) = \mathcal{M}(y_0)$.

Fig. 2 shows the framework of our method. Our method can be divided into two main steps. First, we use the IPM to understand the complex user input $y_k$ and context, generating text prompt $T_k$, edition strength $s_k$, and feedback text $R_k$. The second step centers on generating and editing the character control parameters $\boldsymbol{x}_k$ based on the text prompt $T_k$ and other auxiliary information through our SLPS. We introduce the instruction parsing process of our IPM in Sec. 3.2. The generating and editing process of our SLPS is described in Sec. 3.3 and Sec. 3.4.

### 3.2 Instruction Parsing

The first stage of our framework involves interacting with users and parsing complex user input during dialogue. There are three main objectives: 1) Generating a feedback text $R_k$ for diverse and natural interaction; 2) Extracting accurate text prompts $T_K$ from complex user input, taking into account the dialogue history;

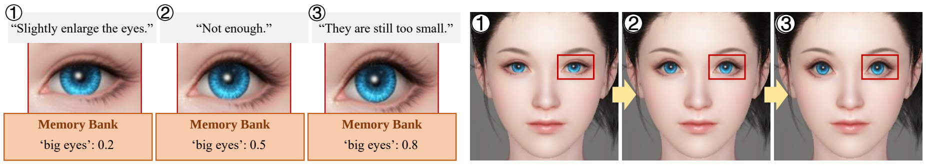


**Fig. 3:** Illustration of editing strength iteratively refining. The character attribute memory bank enables IPM to accurately understand multi-round dialogue and precisely control the editing intensity.

3) Understanding the adjustment intensity $s_k$ of user intention for refining. To achieve these objectives, we introduce LLMs to utilize their powerful interacting and reasoning abilities, and design a character attribute memory bank to track the status of attributes in editing to support players to continuously refine some attributes.

LLMs exhibit an impressive capacity for generalizing to novel samples within a task, given only a limited number of in-context input-output demonstrations. In our approach, we integrate an LLM, prompting it with task-specific background information and a set of diverse examples. This strategy effectively addresses a broad spectrum of user inputs and largely achieves the outlined objectives. By integrating the LLM as a preliminary module, our character editing is adeptly enhanced to effortlessly handle complex and natural user inputs, all without the need for further training.

However, when addressing the need to continuously refine certain attributes, existing LLMs may face issues of hallucination and forgetting. Hence, we design a character attribute memory bank for LLMs that stores and maintains current status of the editing attributes. The editing status mainly includes the editing target on the face, i.e., text prompts, and the corresponding editing intensity, which enables further adjustments by the user.

At the beginning of parsing, IPM constructs an input prompt based on the user input. It then embeds the dialogue history and current editing status into the input prompt, and uses the LLM for parsing. The parsing result includes the text prompt for the editing target $T_k$, editing intensity $s_k$, and system feedback text $R_k$. An example of editing strength refining is shown in Fig. 3. With the help of the LLM and our character attribute memory bank, IPM could understand the editing intent of players even if they use referring expressions, and could further refine the editing strength of the specific editing target. Practically, we employ GPT-4 as our LLM, accessing it through the API of OpenAI. Our prompts are shown in the supplementary material.

### 3.3 Low-dimension Parameter Solving

The second stage of our framework is to use our SLPS to deliver the character control parameters based on the output of IPM. It relies on the basic process of generating character control parameters according to the text prompt. Prior works like T2P [32] offer a solution, but their evolutionary search parameters within an unconstrained space make them slow and unstable, which is unsuitable for an interactive system. In contrast, we propose to optimize parameters within a projected low-dimension space via gradient optimization, enabling swift and reliable generation of character control parameters using text prompts.

The basic pipeline of SLPS uses gradient optimization to find the optimal parameters, which yields an image closest to the text prompt in the pre-trained CLIP embedding space:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} (1 - cos(E_T(T), E_I(G(\boldsymbol{x})))), \tag{2}$$

where $\hat{\boldsymbol{x}}$ is the optimal parameters set that minimizes the cosine distance between the text embedding $E_T(T)$ and the image embedding $E_I(G(\boldsymbol{x}))$. $E_T$ and $E_I$ are the text encoder and image encoder of CLIP, respectively.

To facilitate gradient-based optimization, we employ a neural rendering network imitator $G$ to mimic the rendering process of the game engine that renders characters based on parameters, enabling differentiation throughout the process. In contrast to T2P, our imitator accepts both continuous parameters (e.g., bone position) and discrete parameters (e.g., makeup type) to generate the front view of the game character, bypassing the slow evolutionary discrete parameters search within the game engine.

Directly optimizing $\boldsymbol{x}$ based on a CLIP loss sometimes produces exaggerated or unnatural character representations. To address this, we transition to a projected low-dimension space that conforms to the prior distribution of the characters. As an example, since multiple bones influence eyes of the character, independent parameter shifts can cause skeletal inconsistencies. By adopting dimensionality reduction techniques like PCA [16] or VAE [10] and using a latent code, $\boldsymbol{z} \in \mathbb{R}^M$, we ensure coordinated control across these areas. Hence, our focus shifts to optimizing $\boldsymbol{z}$ rather than $\boldsymbol{x}$. To ensure that the generated characters remain visually coherent, we further integrate a prior distribution constraint, guiding the optimization towards more natural and aesthetically appealing results. Hence, the principle described in Eq. (2) can be expanded as

$$\hat{\boldsymbol{z}} = \arg\min_{\boldsymbol{z}} \mathcal{L}_{CLIP}(T, G(D(\boldsymbol{z}))) + \lambda \mathcal{L}_{Prior}(\boldsymbol{z}), \tag{3}$$

where $\mathcal{L}_{CLIP}$ is the CLIP distance loss described in Eq. (2), and $\boldsymbol{x} = D(\boldsymbol{z})$ denotes the decoder that translates parameters from the reduced representation back to its original form. We adopt a normal prior [1,29] to implement our prior distribution constrain, defined as

$$\mathcal{L}_{Prior}(\boldsymbol{z}) = \|\boldsymbol{A}_{\boldsymbol{z}}(\boldsymbol{z} - \boldsymbol{\mu}_{\boldsymbol{z}})\|^2, \tag{4}$$

where $\boldsymbol{A}_{\boldsymbol{z}}$ and $\boldsymbol{\mu}_{\boldsymbol{z}}$ represent the covariance matrix and mean vector, respectively, derived from a collection of character control parameters.

### 3.4 Fine-grained Parameter Editing

Expanding on the character parameter solving discussed previously, this section aims to enable fine-grained editing of these parameters without unnecessary alterations. A key objective is to semantically align and adjust relevant areas and attributes as specified by the text prompt, while ensuring that unrelated components are preserved. Additionally, modulating the intensity of these edits is an essential capability.

Numerous studies in the relevant domains have provided invaluable insights. Employing regularization to control editing changes is sensitive to hyperparameters, often resulting in insufficient edits or poor preservation of unrelated areas. Another approach leverages the transferability of CLIP embeddings between text and image spaces, as seen in Rodin [27] or StyleCLIP [15]. Yet, the perceived efficacy of this transferability has been overestimated as shown in DeltaEdit [14], leading to less than ideal semantic outcomes in practice.

In our approach, we employ a transformer-based network named the Character Control Parameters Localizer to localize modification-related parameters, which are then optimized by the SLPS in a differentiable manner. The Character Control Parameters Localizer takes the text prompt $T$ as input and performs the multi-class classification, generating semantic labels (e.g., "nose" and "eye-shadow") that indicate modification-related areas and elements. Sequentially, based on the physical interpretation of each character control parameters channel, semantic labels are associated with corresponding channels, culminating in the generation of a binary Character Control Parameters Mask $\boldsymbol{r} \in \mathbb{N}^N$. Each element of $\boldsymbol{r}$ effectively distinguishes between the channels of parameters $\boldsymbol{x}$ that are pertinent or impertinent to the given text prompt. With the mask, we can achieve fine-grained editing by masking channels of parameters during optimization, calculated as

$$\boldsymbol{x}_k = (1 - \boldsymbol{r}) \cdot \hat{\boldsymbol{x}}_{k-1} + \boldsymbol{r} \cdot D(\boldsymbol{z}_k). \tag{5}$$

To train our Character Control Parameters Localizer, we harness ChatGPT to generate 10,000 potential user-editing texts. Initially, ChatGPT assists in performing a coarse categorization of these texts. Thereafter, human annotators meticulously provide fine-grained classification labels. Given that these multi-class labels are heavily unbalanced, we utilize ZLPR loss [25] to address this issue, denoted as

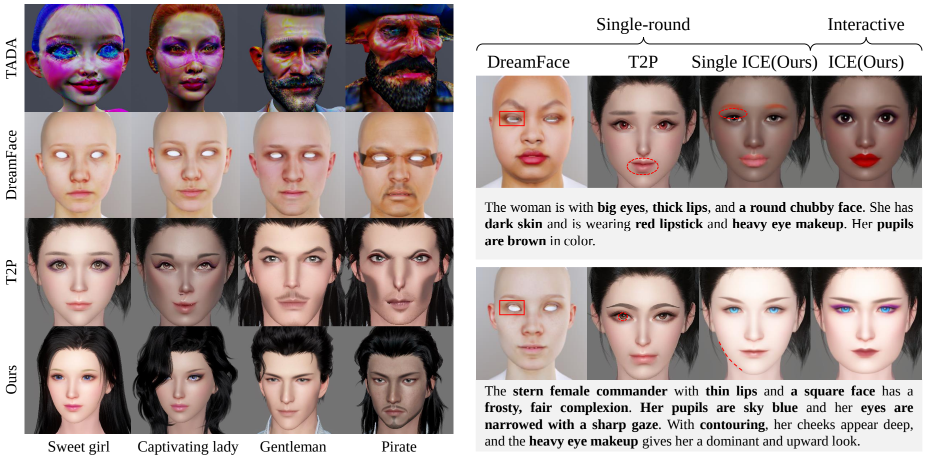
$$\mathcal{L}_{zlpr} = \log(1 + \sum_{i \in \Omega_{neg}} e^{\boldsymbol{s}_i}) + \log(1 + \sum_{i \in \Omega_{pos}} e^{-\boldsymbol{s}_j}), \tag{6}$$

where $\boldsymbol{s}$ is the score vector corresponding to $\boldsymbol{r}$ and $\Omega_{pos}$ is the label set and $\Omega_{neg} = \Lambda / \Omega_{pos}$. We employ RoBERTa [13] for text embedding extraction.

To address the second challenge, controlling the editing intensity is crucial in aligning the final output closely with user intent. The IPM analyzes and deciphers the intended editing strength $s$ based on user intention. It predominantly influences the weight of CLIP loss, denoted as $\lambda_s$, thus effectively modulating the editing strength. In summary, the editing process can be described as

$$\hat{\boldsymbol{z}}_k = \arg\min_{\boldsymbol{z}_k} \lambda_s \mathcal{L}_{CLIP}(T, G(\boldsymbol{x}_k)) + \lambda \mathcal{L}_{Prior}(\boldsymbol{z}_k), \tag{7}$$

where $\boldsymbol{x}_k$ is the mixed parameter described in Eq. (5), and the weight of the CLIP loss is influenced by strength as $\lambda_s = -\cos(s \cdot \pi) + 1$.



The woman is with **big eyes**, **thick lips**, and **a round chubby face**. She has **dark skin** and is wearing **red lipstick** and **heavy eye makeup**. Her **pupils are brown** in color.

The **stern female commander** with **thin lips** and **a square face** has a **frosty, fair complexion**. **Her pupils are sky blue** and her **eyes are narrowed with a sharp gaze**. With **contouring**, her cheeks appear deep, and the **heavy eye makeup** gives her a dominant and upward look.

**(a)** **(b)**

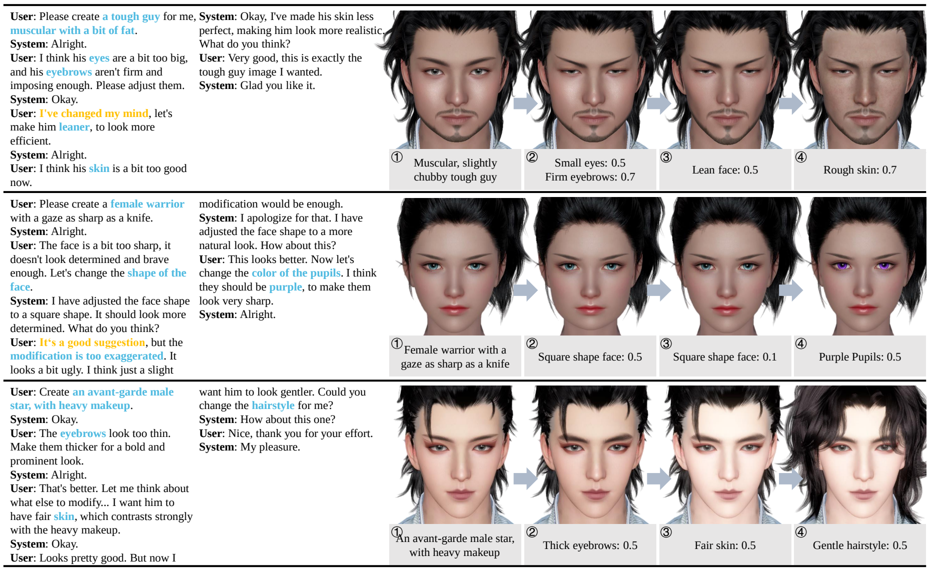
**Fig. 4:** Comparison of final creation. (a) Comparison of our method with state-of-the-art in the single-round creation. In the traditional single-round creation task, our method generates more high-quality results, avoiding abnormal faces, while maintaining strong semantic consistency. (b) Visualization of final character under long text description. Some error attributes are labeled in red. When confronted with extensive text prompts, our full ICE method further maintains fidelity to each detail.

# 4 Experiment

## 4.1 Implementation Details

In this paper, the game characters used are male and female characters from the game *Justice Online Mobile*, a retro-styled RPG game. Character control parameters consist of 450 dimensions, i.e., $x \in \mathbb{R}^{450}$. This includes 284 dimensions of facial bone parameters and 166 dimensions of makeup parameters. The makeup parameters contain 125 discrete parameters, represented by one-hot vectors, which represent different makeup categories. For more information about specific facial bones and makeup parameters, please refer to [32].

**SLPS.** For dimensionality reduction, we simply utilize PCA for facial bone parameters, specifically setting the number of PCA components to 60, while retaining the makeup parameters due to their greater independence. Consequently, the reduced dimensionality amounts to 226, i.e., $\boldsymbol{z} \in \mathbb{R}^{226}$. To extract the prior distributions $\boldsymbol{A}_{\boldsymbol{z}}$ and $\boldsymbol{\mu}_{\boldsymbol{z}}$, we employ an image-driven automatic face-creating algorithm [21] on the publicly available facial dataset CelebA, generating 10,000 character control parameters for each role. Similar to T2P, we pretrain the imitator and CLIP, maintaining our low-dimension character control parameters representation $\boldsymbol{z}$ as the sole variable throughout the gradient optimization process. Our imitator, consistent with [21], comprises eight transposed convolution layers. During the gradient optimization process to identify optimal parameters, we iterate for 100 steps to achieve convergence. We adopt SGD as the optimizer,



**User**: Please create **a tough guy** for me, **muscular with a bit of fat**.
**System**: Alright.
**User**: I think his **eyes** are a bit too big, and his **eyebrows** aren't firm and imposing enough. Please adjust them.
**System**: Okay.
**User**: **I've changed my mind**, let's make him **leaner**, to look more efficient.
**System**: Alright.
**User**: I think his **skin** is a bit too good now.
**System**: Okay, I've made his skin less perfect, making him look more realistic. What do you think?
**User**: Very good, this is exactly the tough guy image I wanted.
**System**: Glad you like it.

① Muscular, slightly chubby tough guy
② Small eyes: 0.5 Firm eyebrows: 0.7
③ Lean face: 0.5
④ Rough skin: 0.7

**User**: Please create a **female warrior** with a gaze as sharp as a knife.
**System**: Alright.
**User**: The face is a bit too sharp, it doesn't look determined and brave enough. Let's change the **shape of the face**.
**System**: I have adjusted the face shape to a square shape. It should look more determined. What do you think?
**User**: **It‘s a good suggestion**, but the **modification is too exaggerated**. It looks a bit ugly. I think just a slight modification would be enough.
**System**: I apologize for that. I have adjusted the face shape to a more natural look. How about this?
**User**: This looks better. Now let's change the **color of the pupils**. I think they should be **purple**, to make them look very sharp.
**System**: Alright.

**User**: Create **an avant-garde male star, with heavy makeup**.
**System**: Okay.
**User**: The **eyebrows** look too thin. Make them thicker for a bold and prominent look.
**System**: Alright.
**User**: That's better. Let me think about what else to modify... I want him to have fair **skin**, which contrasts strongly with the heavy makeup.
**System**: Okay.
**User**: Looks pretty good. But now I want him to look gentler. Could you change the **hairstyle** for me?
**System**: How about this one?
**User**: Nice, thank you for your effort.
**System**: My pleasure.

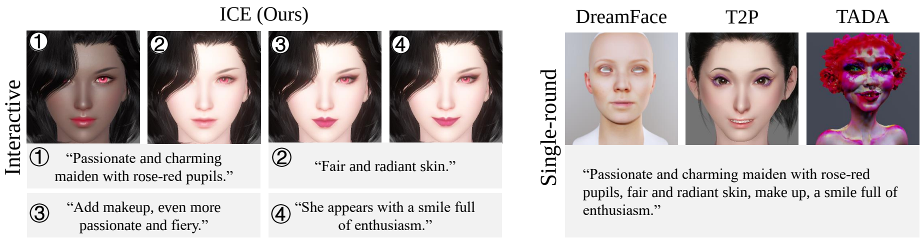
**Fig. 5:** Visualization of interactive character editing process with our proposed method. Important modification instructions are highlighted in blue, while inspirations derived by players from the system are marked in yellow. The parsed instructions along with their corresponding intensity levels during each edit are presented in a gray box.

setting a learning rate of 1.0 for continuous parameters and 100.0 for discrete parameters. The prior loss weight $\lambda$ is established at 8e-4. For editing tasks, the initial optimization value is the low-dimension variable $\boldsymbol{z_{t-1}}$ corresponding to the parameters from the preceding step; for initial creation, the starting value is the mean of the prior distribution $\mu_{\boldsymbol{z}}$.

**Character Control Parameters Localizer.** The Character Control Parameters Localizer is composed of a RoBERTa [13] model followed by a linear layer. We initialize our model with the pre-trained weights of "roberta-large", which features 24 hidden layers, 16 attention heads per layer, and a hidden size of 1024. During the training phase, we optimize all model weights using the AdamW optimizer, with a batch size of 64 and a learning rate of 3e-5. Our dataset includes 9,800 instances, of which 20% were designated as a validation set, with the remaining data used for training. Each text in the training set is associated with labels for 117 categories.

### 4.2 Qualitative Evaluation

We conduct a qualitative comparison of our ICE framework with established methods: DreamFace [31], T2P [32], and TADA [12]. We evaluate these methods for both the traditional single-round creation task and our proposed interactive multi-round editing process.



① "Passionate and charming maiden with rose-red pupils."

② "Fair and radiant skin."

③ "Add makeup, even more passionate and fiery."

④ "She appears with a smile full of enthusiasm."

"Passionate and charming maiden with rose-red pupils, fair and radiant skin, make up, a smile full of enthusiasm."

**Fig. 6:** Comparison between our method and state-of-the-arts. Our method enables interactive character editing, whereas prevalent methods can only directly generate characters in a single round based on a comprehensive description. Beyond improving interaction, it also addresses the inaccuracies and unreliability observed in the outcomes of existing methods.

**Character creation comparison.** As shown in Fig. 4a and Fig. 4b, we evaluate the final character creation outcomes of prevalent methods and ours. Initially, characters are created based on single, brief text prompts in a single-round manner, comparing them in Fig. 4a. Although TADA maintains semantic consistency, it yields odd outcomes due to its direct generation of textures and geometries. T2P generates character control parameters of *Justice Online Mobile*, but optimizes the raw parameters directly, also resulting in abnormal faces. Dreamface results lack distinct consistency with textual descriptions. By solving parameters in a low-dimension space, our method outperforms existing methods in quality, effectively avoiding abnormal faces, while ensuring strong semantic consistency. Furthermore, when handling extensive text prompts, as shown in Fig. 4b, all single-round creation methods showed discrepancies, diverging in certain attributes from the textual descriptions. However, our ICE method maintains fidelity to textual descriptions in every detail, highlighting the superiority of our multi-round editing approach.

**Interaction process presentation.** Several illustrative cases of our interactive character editing process are presented in Fig. 5. These examples demonstrate the capability of our framework of diverse and fine-grained control over character parameter editing through interactive dialogue. This process consistently generates high-quality characters initially, and permits iterative, fine-grained modifications without affecting unrelated areas. Additionally, it efficiently tracks editing status of the character, enabling accurate and easy iterative refinement of attributes and their intensities. Our framework significantly enhances the user experience by facilitating a natural and comprehensive dialogue interaction. Players can not only ensure that the results meet their preferences through iterative adjustments but can also, as demonstrated in the examples, be inspired and generate new ideas during the dialogue and editing process.

**Interaction comparison.** In Fig. 6, the ICE framework is compared to prevalent single-round creation methods To the best of our knowledge, this is the first work focusing on interactive 3D game character editing. Referenced methods primarily use single-round creation, generating characters from a single com-

**Table 1:** Quantitative evaluation of our method and the state-of-the-art.

| Methods | Objective Evaluations | | Subjective Evaluations | | |
|---|---|---|---|---|---|
| | CLIP score | Running time | Consistency with text | Quality | Preference |
| DreamFace | 0.2362 | >300s | 1.553 | 1.777 | 2.5% |
| TADA | 0.2689 | 4.5h | 1.937 | 1.882 | 13.0% |
| T2P | 0.2480 | 359.47s | 2.066 | 2.089 | 7.4% |
| ICE | **0.2699** | **5.70s + 3.34s** | **3.756** | **4.061** | **77.1%** |

prehensive textual prompt. Additionally, these methods lack the capability for further adjustments if outcomes are unsatisfactory. In contrast, the ICE framework allows for interactive character editing until it aligns with user vision. For comparison, the interactive editing process is approximated by concatenating and modifying text prompts for these methods, as demonstrated in [9, 12]. Details of this comparison are included in the supplementary material.

### 4.3 Quantitative Evaluation

Our method is quantitatively compared with previous methods, DreamFace, T2P, and TADA, through objective and subjective evaluations. Ten different text prompts are fed into these methods and our proposed ICE to generate characters.

**Objective evaluation.** Following previous works, we calculate the CLIP score by computing the cosine similarity of image features and text features and measure the response time of each method, as shown in Tab. 1. Except for DreamFace, all methods are executed on an NVIDIA A30 GPU. Due to DreamFace not being open-sourced, its reported time on an NVIDIA A6000 is referenced, which is expected to be longer on the A30. Given the multi-round interactive nature of our method, the running time for responding to user input per round is presented. This includes the time taken to request the GPT-4 API, averaging around 5.70 seconds in our case, which may vary based on the language model used and network latency. The proposed ICE responds much faster, not only enhancing performance in traditional single-round creation tasks, but also facilitating quicker feedback during interactive editing. Moreover, ICE achieves a higher CLIP score compared to other methods, indicating superior semantic consistency between the results and textual descriptions. Among the competitors, TADA secures the second-highest score, consistent with its subjective assessment of demonstrating high semantic consistency albeit with lower quality.

**Subjective evaluation.** We conducted an extensive user study involving 100 participants to assess the quality and text consistency of the generated character results. Participants were asked to rate the heads of characters on a scale from 1 to 5. The description of each score is presented in the supplementary material. Furthermore, participants were asked to select their preferred results among those generated by DreamFace, TADA, T2P, and our ICE method. As indicated in Tab. 1, our method not only achieves high scores in quality and consistency, but also emerges as the most preferred among participants.

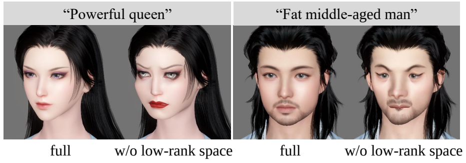


**Fig. 7:** Ablation on low-dimension space optimization in our SLPS. Optimizing raw character control parameters without projecting them into a low-dimension space leads to unrealistic face shapes.

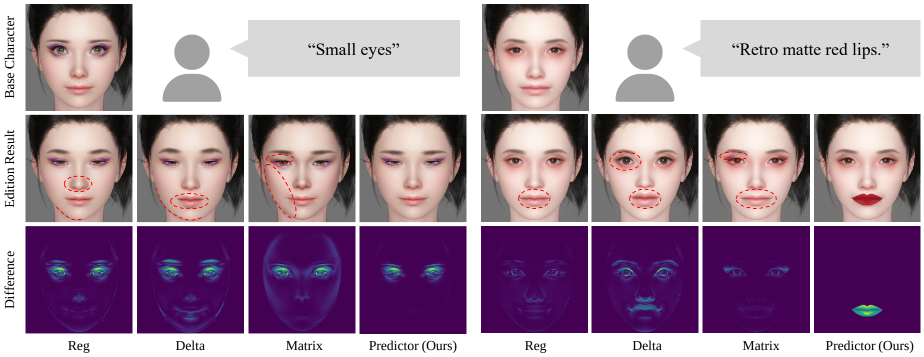


**Fig. 8:** Ablation on editing implementation. In contrast to other naive methods, which may inadvertently alter unrelated regions or face challenges in achieving semantic edits, our approach exhibits consistent precision and reliability.

## 4.4 Ablation Study

**Ablation on low-dimension space optimization.** As demonstrated in Fig. 7, optimizing raw character control parameters without low-dimension space projection leads to unrealistic facial shape creation. Our method optimizes parameters in a low-dimension space, ensuring the generated results on a Grassmann manifold.

**Ablation on editing implementation.** To further validate the effectiveness of our editing method, comparisons were drawn with several naive editing baselines: 1) **Reg**: Applying regularization on images or parameters; 2) **Delta**: Utilizing delta text embedding to derive the editing direction; 3) **Matrix**: Calculating a relevance matrix to establish correlations between clip embedding and parameter channels. Additional details of these baselines are provided in the supplementary materials. Fig. 8 reveals that while these methods either unintentionally influence unrelated regions or falter in effecting semantic edits, our editing approach remains consistently precise and reliable.

**Ablation on memory bank.** The comparison between the editing process utilizing IPM with and without the memory bank is depicted in Fig. 9. For each round, user input, parsed instructions, and the corresponding generated character are showcased. The results indicate that without the integration of a character attributes memory bank, the LLM tends to inaccurately predict editing intensity during the refinement process.

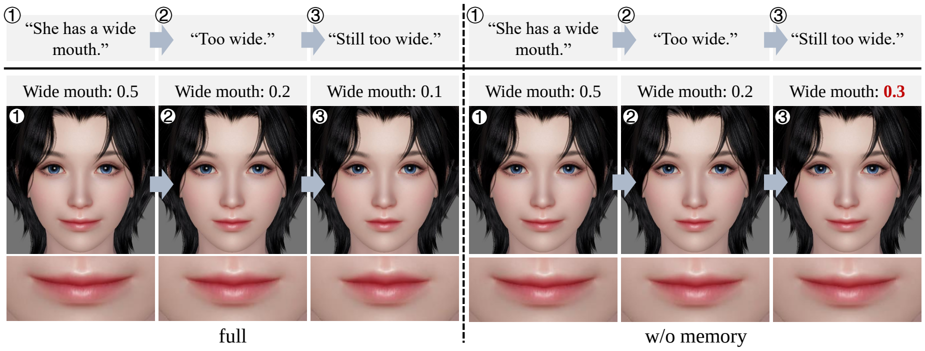


**Fig. 9:** Ablation on character attribute memory bank in our IPM. Without a memory bank, the LLM struggles to accurately determine the editing intensity during the refinement process.

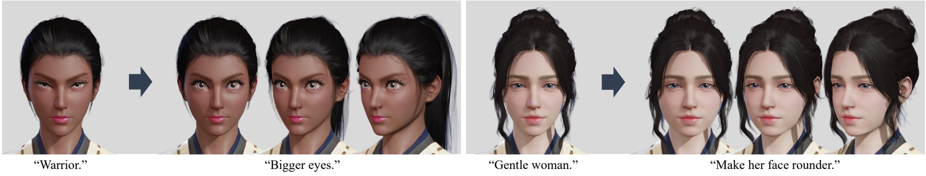


**Fig. 10:** Examples of our method in the game of Naraka: Bladepoint. Our method can easily extend to other games.

**Results on Other Games.** We test our method in another game, Naraka: Bladepoint, as shown in Fig. 10. This demonstrates the adaptability to support various games of our method. For new game adaption, only the imitator is retrained to mimic the new game rendering process, without any other networks training. Character control parameter localization requires merely aligning semantic labels with channels according to their physical interpretation in the new game, thus bypassing the need to retrain the Localizer.

## 5 Conclusion

This work introduced the Interactive Character Editing (ICE) framework, which achieves a multi-round, dialogue-based 3D game character refinement process. Unlike traditional single-round generation systems, ICE provides a user-friendly way that enables players to convey creative ideas iteratively while ensuring that created characters align with the expectations of players. Designed for game systems, ICE reliably and swiftly applies instructions, and allows for seamless integration into existing systems with minimal effort. Experimental validations have demonstrated robustness, precision, and superior performance of ICE. Despite setting new benchmarks, the ICE still exhibits limitations, notably in the speed of parameter solving through iterative optimization. Future efforts will focus on enhancing response speed of the system.

## References

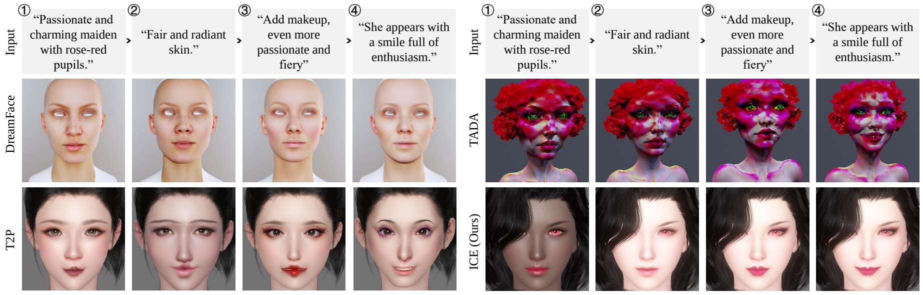
1. Agarwal, S., Mierle, K., Team, T.C.S.: Ceres Solver (10 2023), `https://github.com/ceres-solver/ceres-solver` 7
2. Borovikov, I., Levonyan, K., Rein, J., Wrotek, P., Victor, N.: Applied monocular reconstruction of parametric faces with domain engineering. arXiv preprint arXiv:2208.02935 (2022) 2, 3
3. Cao, Y., Cao, Y.P., Han, K., Shan, Y., Wong, K.Y.K.: Dreamavatar: Text-and-shape guided 3d human avatar generation via diffusion models. arXiv preprint arXiv:2304.00916 (2023) 4
4. Cui, X., Li, Z., Li, P., Hu, Y., Shi, H., He, Z.: I2edit: Towards multi-turn interactive image editing via dialogue. arXiv preprint arXiv:2303.11108 (2023) 4
5. El-Nouby, A., Sharma, S., Schulz, H., Hjelm, D., Asri, L.E., Kahou, S.E., Bengio, Y., Taylor, G.W.: Tell, draw, and repeat: Generating and modifying images based on continual linguistic instruction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10304–10312 (2019) 4
6. Han, X., Cao, Y., Han, K., Zhu, X., Deng, J., Song, Y.Z., Xiang, T., Wong, K.Y.K.: Headsculpt: Crafting 3d head avatars with text. arXiv preprint arXiv:2306.03038 (2023) 4
7. Hong, F., Zhang, M., Pan, L., Cai, Z., Yang, L., Liu, Z.: Avatarclip: Zero-shot text-driven generation and animation of 3d avatars. arXiv preprint arXiv:2205.08535 (2022) 2, 4
8. Jiang, Y., Huang, Z., Pan, X., Loy, C.C., Liu, Z.: Talk-to-edit: Fine-grained facial editing via dialog. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 13799–13808 (2021) 4
9. Joseph, K., Udhayanan, P., Shukla, T., Agarwal, A., Karanam, S., Goswami, K., Srinivasan, B.V.: Iterative multi-granular image editing using diffusion models. arXiv preprint arXiv:2309.00613 (2023) 4, 12, 17
10. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013) 7
11. Kolotouros, N., Alldieck, T., Zanfir, A., Bazavan, E.G., Fieraru, M., Sminchisescu, C.: Dreamhuman: Animatable 3d avatars from text. arXiv preprint arXiv:2306.09329 (2023) 4
12. Liao, T., Yi, H., Xiu, Y., Tang, J., Huang, Y., Thies, J., Black, M.J.: Tada! text to animatable digital avatars. arXiv preprint arXiv:2308.10899 (2023) 4, 10, 12, 17
13. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019) 8, 10
14. Lyu, Y., Lin, T., Li, F., He, D., Dong, J., Tan, T.: Deltaedit: Exploring text-free training for text-driven image manipulation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6894–6903 (2023) 4, 8, 20
15. Patashnik, O., Wu, Z., Shechtman, E., Cohen-Or, D., Lischinski, D.: Styleclip: Text-driven manipulation of stylegan imagery. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2085–2094 (2021) 4, 8, 18, 20
16. Pearson, K.: Liii. on lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin philosophical magazine and journal of science **2**(11), 559–572 (1901) 7
17. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: Text-to-3d using 2d diffusion. arXiv preprint arXiv:2209.14988 (2022) 4

18. Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H.: Generative adversarial text to image synthesis. In: International conference on machine learning. pp. 1060–1069. PMLR (2016) 4
19. Revanur, A., Basu, D., Agrawal, S., Agarwal, D., Pai, D.: Coralstyleclip: Co-optimized region and layer selection for image editing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12695–12704 (2023) 4
20. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10684–10695 (2022) 4
21. Shi, T., Yuan, Y., Fan, C., Zou, Z., Shi, Z., Liu, Y.: Face-to-parameter translation for game character auto-creation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 161–170 (2019) 2, 3, 9
22. Shi, T., Zou, Z., Shi, Z., Yuan, Y.: Neural rendering for game character auto-creation. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(3), 1489–1502 (2020) 2, 3
23. Shi, T., Zou, Z., Song, X., Song, Z., Gu, C., Fan, C., Yuan, Y.: Neutral face game character auto-creation via pokerface-gan. In: Proceedings of the 28th ACM International Conference on Multimedia. pp. 3201–3209 (2020) 2, 3
24. Shi, T., Zuo, Z., Yuan, Y., Fan, C.: Fast and robust face-to-parameter translation for game character auto-creation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 1733–1740 (2020) 2, 3
25. Su, J., Zhu, M., Murtadha, A., Pan, S., Wen, B., Liu, Y.: Zlpr: A novel loss for multi-label classification (2022) 8
26. Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. arXiv preprint arXiv:2106.10689 (2021) 4
27. Wang, T., Zhang, B., Zhang, T., Gu, S., Bao, J., Baltrusaitis, T., Shen, J., Chen, D., Wen, F., Chen, Q., et al.: Rodin: A generative model for sculpting 3d digital avatars using diffusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4563–4573 (2023) 4, 8, 18
28. Wolf, L., Taigman, Y., Polyak, A.: Unsupervised creation of parameterized avatars. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1530–1538 (2017) 2, 3
29. Xiang, D., Joo, H., Sheikh, Y.: Monocular total capture: Posing face, body, and hands in the wild. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10965–10974 (2019) 7
30. Yue, D., Guo, Q., Ning, M., Cui, J., Zhu, Y., Yuan, L.: Chatface: Chat-guided real face editing via diffusion latent space manipulation. arXiv preprint arXiv:2305.14742 (2023) 4
31. Zhang, L., Qiu, Q., Lin, H., Zhang, Q., Shi, C., Yang, W., Shi, Y., Yang, S., Xu, L., Yu, J.: Dreamface: Progressive generation of animatable 3d faces under text guidance. arXiv preprint arXiv:2304.03117 (2023) 2, 4, 10
32. Zhao, R., Li, W., Hu, Z., Li, L., Zou, Z., Shi, Z., Fan, C.: Zero-shot text-to-parameter translation for game character auto-creation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 21013–21023 (2023) 2, 4, 7, 9, 10
33. Zhou, Y., Zhang, R., Gu, J., Tensmeyer, C., Yu, T., Chen, C., Xu, J., Sun, T.: Tigan: Text-based interactive image generation and manipulation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 3580–3588 (2022) 4

# A Additional Experiments

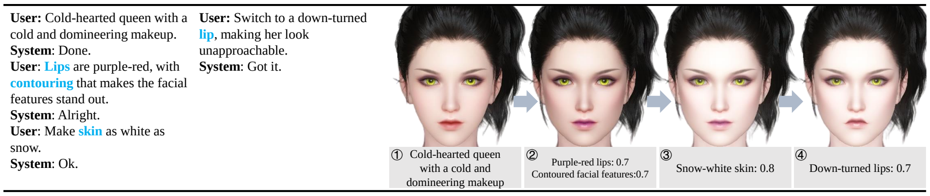
## A.1 Comparison of Interactive Editing Process

As mentioned in Sec. 4.1, we approximate the interactive editing process of them by concatenating and modifying text prompts for comparison, as demonstrated in [9, 12]. The results are shown in Fig. A1. Simply editing characters through text prompt adjustments often lead to global modifications and fail in achieving iterative fine-grained editing. The approximated editing done by previous methods often fails to preserve the integrity of previous edits, resulting in unexpected drastic changes across iterations. Furthermore, these methods occasionally overlook certain attributes, especially with elongated text prompts in the last rounds. In contrast, our ICE framework could fine-grained alter semantically related attributes, avoiding unnecessary modifications.



① "Passionate and charming maiden with rose-red pupils." ② "Fair and radiant skin." ③ "Add makeup, even more passionate and fiery" ④ "She appears with a smile full of enthusiasm."

**Fig. A1:** Comparison between our method and approximated interactive editing process by state-of-the-arts. Text prompts are concatenated and modified in each round to approximate the interactive editing process for the single-round methods.



**User:** Cold-hearted queen with a cold and domineering makeup.
**System**: Done.
**User**: **Lips** are purple-red, with **contouring** that makes the facial features stand out.
**System**: Alright.
**User**: Make **skin** as white as snow.
**System**: Ok.
**User:** Switch to a down-turned **lip**, making her look unapproachable.
**System**: Got it.

① Cold-hearted queen with a cold and domineering makeup
② Purple-red lips: 0.7 Contoured facial features:0.7
③ Snow-white skin: 0.8
④ Down-turned lips: 0.7

**Fig. A2:** Results obtained using Claude 3 as our LLM.

## A.2 Employing Alternative LLMs

Our approach is compatible with alternative LLMs, not limited to GPT-4. As illustrated in Fig. A2, Our framework remains effective when utilizing Claude 3 as our LLM.

### A.3 More Cases

More cases of interactive character editing process of our proposed method are shown in Fig. A3.

## B Experimental Details

### B.1 Score Description of User Study

In our user study, participants were asked to rate the heads of characters on a scale from 1 to 5. **The quality score** ranged from 1 to 5, with 1 being "extremely ugly and non-human-like", 2 as "slightly flawed, needs improvement", 3 as "acceptable, barely satisfactory", 4 as "quite good, only a few areas need refinement", to 5 being "aesthetically pleasing and natural". For **consistency with the text**, the scores ranged from 1 to 5, where 1 represented "no relation at all", 2 as "ambiguous", 3 as "reasonable, generally matches", 4 as "very similar, mostly conforms", to 5 indicating "perfectly consistent".

### B.2 Naive Edition Baselines

We conduct a comprehensive comparison of our proposed method against several fundamental editing baselines, as visually depicted in Fig. 9. Below, we detail these baseline methodologies:

**Reg.** Similar to the approach described in [15], this baseline applies regularization to either images or parameters, aiming to preserve irrelevant attributes from being altered. The process of regularization on images is mathematically formulated as

$$\begin{aligned}\hat{\boldsymbol{z}}_k = \arg\min_{\boldsymbol{z}_k} & \mathcal{L}_{CLIP}(T, G(D(\boldsymbol{z}_k))) + \lambda \mathcal{L}_{Prior}(\boldsymbol{z}_k) \\ & + \lambda_r \|G(D(\boldsymbol{z}_k)) - G(D(\boldsymbol{z}_{k-1}))\|^2,\end{aligned} \tag{8}$$

and regularization on parameters is described as

$$\begin{aligned}\hat{\boldsymbol{z}}_k = \arg\min_{\boldsymbol{z}_k} & \mathcal{L}_{CLIP}(T, G(D(\boldsymbol{z}_k))) + \lambda \mathcal{L}_{Prior}(\boldsymbol{z}_k) \\ & + \lambda_r \|\boldsymbol{z}_k - \boldsymbol{z}_{k-1}\|^2.\end{aligned} \tag{9}$$

Selecting an appropriate value for $\lambda_r$ is crucial, yet challenging. Setting $\lambda_r$ too high can hinder necessary modifications, while a too low value might lead to unwanted changes in irrelevant areas.

**Delta.** Similar to the concept presented in [27], the core principle of this method involves deriving the editing direction utilizing delta text embedding. The delta text embedding, denoted as $\boldsymbol{\delta}$, is obtained through prompt engineering, exemplified by the following equation,

$$\boldsymbol{\delta} = E_T(T) - E_T('a\ human\ face'), \tag{10}$$

**User**: I haven't decided what kind of character to create yet. Create a **cute girl** for me.
**System**: Sure.
**User**: Let me think… Let's change the shape of the **mouth**. Make it narrow with thick lips, like plump cherries.
**System**: Okay.
**User**: I have an idea! I want to create a **gothic girl**! What else do you think is missing?
**System**: Gothic girls **often have black eyeshadow and eyeliner, as well as dark lipstick.** What do you think about that?
**User**: I think that's great! Let's go with that!
**System**: Alright.
**User**: Let's find a **hairstyle** that suits her.
**System**: Sure, we can try **long straight black hair**. What do you think?
**User**: Great! I like her.

**User**: I'm envisioning **a dominant commander** character.
**System**: Sure.
**User**: Hmm, it's not quite what I had in mind. I'm actually picturing a more mature-looking commander. How about adding a **stubble beard**?
**System**: Got it.
**User**: Perfect! And for a battle-hardened look, let's go with **dark, sun-tanned skin**.
**System**: I've made the update.
**User**: Nice, but let's make the **face rounder**.
**System**: Done.

**User**: I haven't decided what kind of character to create. Please create a **gentle woman** for me to start with.
**System**: Sure.
**User**: I would like her to have a more eye-catching appearance. Please add some **makeup** for me.
**System**: Alright.
**User**: Let's make her **lips** a bit thicker.
**System**: Understood.
**User**: I hope she can have a rounder **face**, as it would make her look even more gentle.
**System**: Got it.
**User**: I think that's about it. Thank you for your cooperation.
**System**: Alright, I believe we've achieved the desired look.

**User**: Create a **pretty boy** character for a game.
**System**: Done.
**User**: Make the **skin** a bit darker to look healthier.
**System**: Alright.
**User**: I'm not too fond of this middle-parted **hairstyle**. Are there other options?
**System**: Certainly, **we have many other hairstyle options** like short, long, curly, straight, etc. What type of hairstyle would you prefer?
**User**: I'd like a messy, curly hairstyle.
**System**: Okay, how does it seem?
**User**: Looks good. Finally, let's adjust the **face** shape. The chin is too pointed and long, it looks a bit odd.
**System**: Okay.

**User**: Create a **lively girl** for me.
**System**: Sure.
**User**: Could you make the **skin** look more rosy?
**System**: Of course.
**User**: The **face** doesn't look cute enough. Can you make it rounder and chubby?
**System**: Absolutely.
**User**: Change the **hairstyle** to double buns, please.
**System**: Certainly.
**User**: I think it's good now. Let's keep it like this. Thank you.
**System**: I'm delighted that you're satisfied. If there are any other adjustments you'd like, feel free to let me know.

**User**: I'm looking to create a character of a **mysterious female witch**.
**System**: Sure.
**User**: Could her **eyes** be slightly larger? I think it would enhance her mystique.
**System**: Of course.
**User**: They **still seem a bit small** to me.
**System**: I've increased the size further. How about now?
**User**: Also, changing the color of her **pupils** might add to her mysterious aura. Any suggestions for the color?
**System**: How about we **try purple or green**? These colors often add a touch of mystery.
**User**: Emerald green sounds perfect. Could you make that change?
**System**: Absolutely.

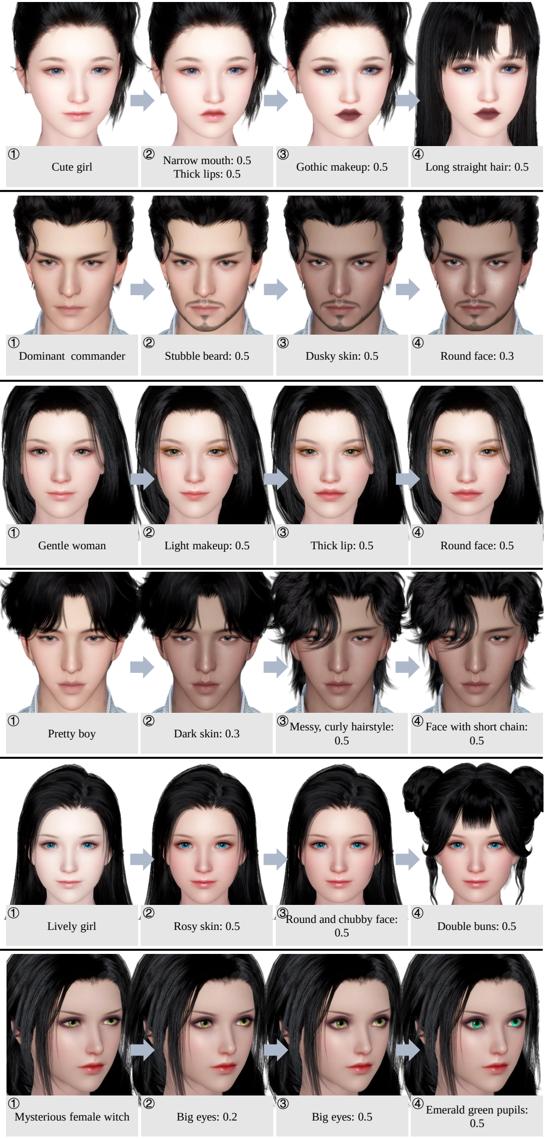


**Fig. A3:** More cases of interactive character editing results with our proposed method.

where $E_T$ is the text encoder of CLIP. By assuming colinearity between the image and text embedding of CLIP, the approach determines the editing direction by applying $\boldsymbol{\delta}$ to the image embedding of the character from the previous round. The entire process is formulated as

$$\begin{aligned}\hat{\boldsymbol{z}}_k = &\arg\min_{\boldsymbol{z}_k}(1-\cos(\boldsymbol{e}_{k-1}+\boldsymbol{\delta}, G(D(\boldsymbol{z}_k)))) \\ &+\lambda\mathcal{L}_{Prior}(\boldsymbol{z}_k),\end{aligned} \tag{11}$$

where $\boldsymbol{e}_{k-1} = G(D(\boldsymbol{z}_{k-1}))$ represents the image embedding of the character from the last iteration. However, as noted in [14], the assumed colinearity between image and text embeddings in CLIP is often overestimated. This overestimation leads to inaccuracies in the semantic direction of editing, as illustrated in Fig. 9.
**Matrix.** Similar to [15], this baseline calculates a relevance matrix to establish channelwise relevance between clip embedding and facial parameters. We first randomly generate a set of facial parameters $\boldsymbol{x}_i \in \mathbb{R}^N$. Subsequently, we apply perturbations to each channel of the parameters in succession, and then calculate the corresponding image of the character along with the changes in respective CLIP embeddings. Let $c$ denote the channel number to which the perturbation is applied, $\boldsymbol{\epsilon}^c$ represent the perturbations and $\Delta\boldsymbol{e}_i^c \in \mathbb{R}^D$ represent the changes in the corresponding CLIP embedding. By averaging over the collection, we obtain the mean CLIP embedding change $\Delta\bar{\boldsymbol{e}}^c$ associated with that particular perturbation. This leads to the formation of a relevance matrix

$$\boldsymbol{R} \in \mathbb{R}^{N\times D},\ where\ \boldsymbol{R}[c] = \Delta\bar{\boldsymbol{e}}^c. \tag{12}$$

At manipulations, given a text prompt, we first obtain the delta text embedding $\boldsymbol{\delta}$ by prompt engineering as described in Eq. (10). Then, assuming the colinearity between image and text embeddings in CLIP, this approach calculates parameter relevance vector as

$$\boldsymbol{r} = \max(|\boldsymbol{\delta}\boldsymbol{R}^T|, \xi) \tag{13}$$

, where $\xi$ is a threshold of relevance. This approach also overestimates the colinearity between image and text embeddings in CLIP and often fails in the semantic direction of editing.

## C IPM Implementation Details

Our prompts are shown in Fig. A4 and Fig. A5. In each request to the LLM, the "*#state#*" and "*#history#*" parts of the prompt are replaced with values from our memory, and the "*#command#*" section is replaced with user input.

# Background Information

There is now a game character editing system, where facial features, makeup, and other appearance attributes can be edited, allowing users to create their favorite game characters. Users can control how to edit characters using natural language.

# Requirements

Based on the user's instruction input, conversation history, and the current state of the character being edited, output the corresponding editing target and strength, and respond accordingly. If the user's instruction contains multiple sub-instructions, please split them. The default editing strength is 0.5, with a maximum of 1 and a minimum of 0. Adjust the editing strength based on the user's tone of speech and the conversation history. Sometimes, users may not provide a clear editing target. In such cases, infer the editing target based on the conversation history and the current state of the character being edited, and confirm with the user before proceeding.

# Additional Information

Output format requirements: Output in JSON, without any redundant information. For example, *{'result': [{'target': 'thick lips', 'strength': 0.5}, {'target': 'smoky makeup', 'strength': 0.5}], 'response': 'Okay, I have changed to thick lips and smoky makeup.'}* In some cases, you will only engage in conversation without making any edits. In such cases, your output should be empty, like *{'result': [], 'response': 'I plan to make the skin color a bit fairer, which should make her look more gentle. What do you think?'}*

# In-context Examples

## Example One

```
input: {
 'instruction': 'Not dark enough',
 'history': [
  'user': 'I want a wheat-colored skin',
  'system': 'Received.'
 ],
 'state': {
  'Wheat-colored skin': 0.5
 }
}
output: {
 'result': [{'target': 'Wheat-colored skin', 'strength': 0.8}],
 'response': 'Okay, a bit darker, how about this?'
}
```

## Example Two

```
input: {
 'instruction': 'A bit whiter',
 'history': [
  'user': 'I want a wheat-colored skin',
  'system': 'Received.'
 ],
 'state': {
  'Wheat-colored skin': 0.5
 }
}
output: {
 'result': [{'target': 'Wheat-colored skin', 'strength': 0.2}],
 'response': How about this?'
}
```

## Example Three

```
input: {
 'instruction': 'Make eyes slightly bigger',
 'history': []
 'state': {}
}
output: {
 'result': [{'target': 'Big eyes', 'strength': 0.2}],
 'response': 'Done.'
}
```

## Example Four

```
input: {
 'instruction': 'Thick eye makeup, and eyebrows with a spirited look',
 'history': []
 'state': {}
}
output: {
 'result': [{'target': 'Thick eye makeup', 'strength': 0.5}, {'target': 'Spirit
ed eyebrows', 'strength': 0.7}],
 'response': 'Adjusted.'
}
```

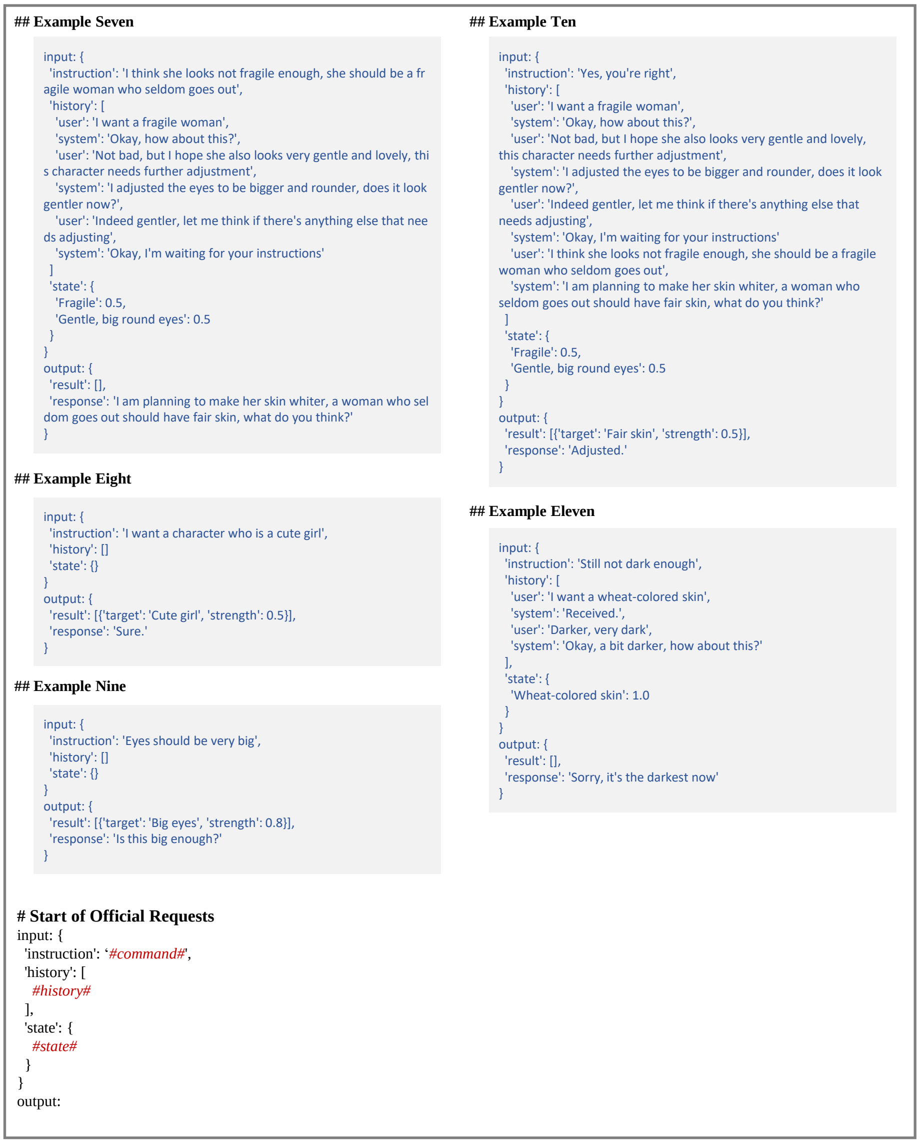
## Example Five

```
input: {
 'instruction': 'Adjust the eye and lip makeup to look sexier',
 'history': []
 'state': {}
}
output: {
 'result': [{'target': 'Sexier eye and lip makeup', 'strength': 0.5}],
 'response': 'Adjusted. '
}
```

## Example Six

```
input: {
 'instruction': 'It looks really nice',
 'history': [
  'user': 'Big eyes',
  'system': 'Done.'
 ]
 'state': {
  'Big eyes': 0.5
 }
}
output: {
 'result': [],
 'response': 'Yes, I think it looks very good too.'
}
```

**Fig. A4:** The brief of our prompt for LLM (a).



## Example Seven

```
input: {
 'instruction': 'I think she looks not fragile enough, she should be a fr
agile woman who seldom goes out',
 'history': [
   'user': 'I want a fragile woman',
   'system': 'Okay, how about this?',
   'user': 'Not bad, but I hope she also looks very gentle and lovely, thi
s character needs further adjustment',
   'system': 'I adjusted the eyes to be bigger and rounder, does it look
gentler now?',
   'user': 'Indeed gentler, let me think if there's anything else that nee
ds adjusting',
   'system': 'Okay, I'm waiting for your instructions'
 ]
 'state': {
   'Fragile': 0.5,
   'Gentle, big round eyes': 0.5
 }
}
output: {
 'result': [],
 'response': 'I am planning to make her skin whiter, a woman who sel
dom goes out should have fair skin, what do you think?'
}
```

## Example Eight

```
input: {
 'instruction': 'I want a character who is a cute girl',
 'history': []
 'state': {}
}
output: {
 'result': [{'target': 'Cute girl', 'strength': 0.5}],
 'response': 'Sure.'
}
```

## Example Nine

```
input: {
 'instruction': 'Eyes should be very big',
 'history': []
 'state': {}
}
output: {
 'result': [{'target': 'Big eyes', 'strength': 0.8}],
 'response': 'Is this big enough?'
}
```

## Example Ten

```
input: {
 'instruction': 'Yes, you're right',
 'history': [
   'user': 'I want a fragile woman',
   'system': 'Okay, how about this?',
   'user': 'Not bad, but I hope she also looks very gentle and lovely,
this character needs further adjustment',
   'system': 'I adjusted the eyes to be bigger and rounder, does it look
gentler now?',
   'user': 'Indeed gentler, let me think if there's anything else that
needs adjusting',
   'system': 'Okay, I'm waiting for your instructions'
   'user': 'I think she looks not fragile enough, she should be a fragile
woman who seldom goes out',
   'system': 'I am planning to make her skin whiter, a woman who
seldom goes out should have fair skin, what do you think?'
 ]
 'state': {
   'Fragile': 0.5,
   'Gentle, big round eyes': 0.5
 }
}
output: {
 'result': [{'target': 'Fair skin', 'strength': 0.5}],
 'response': 'Adjusted.'
}
```

## Example Eleven

```
input: {
 'instruction': 'Still not dark enough',
 'history': [
   'user': 'I want a wheat-colored skin',
   'system': 'Received.',
   'user': 'Darker, very dark',
   'system': 'Okay, a bit darker, how about this?'
 ],
 'state': {
   'Wheat-colored skin': 1.0
 }
}
output: {
 'result': [],
 'response': 'Sorry, it's the darkest now'
}
```

# Start of Official Requests

```
input: {
 'instruction': ‘#command#',
 'history': [
   #history#
 ],
 'state': {
   #state#
 }
}
output:
```

**Fig. A5:** The brief of our prompt for LLM (b).