Computers & Graphics (2024)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Improving Digital Communication with Personalized Expressive Characters in Interactive Comic Scenes

Alexander Schier[a,1], Caro Schmitz[a,1], Reinhard Klein[a]

[a]*Rheinische Friedrich-Wilhelms-Universität Bonn, Institute of Computer Science II, Friedrich-Hirzebruch-Allee 8, 53113 Bonn, Germany*

ARTICLE INFO

Article history:
Received August 30, 2024

Keywords: Gen-AI, Stable Diffusion, image generation, non-photorealistic rendering, illustration, comics, comic generation, visual storytelling, graphical text representation, chat programs, world wide web

ABSTRACT

We introduce an interactive comic chat system that seamlessly integrates visual storytelling with text-based real-time interaction to enhance digital communication. Our approach employs generative AI to create interactive comics, supporting the easy creation of personalised characters and scenes, dynamically adapting these personalised characters and environments to chat content and additional user input, automating the generation of visually coherent comic panels, and overcoming the limitations of previous comic chat systems that are limited to pre-defined graphical elements. We propose algorithms for generating characters and panel backgrounds using generative AI, incorporating facial expressions, poses and thematic elements, and effectively placing characters and speech bubbles in comic panels. The continuous changes in character details and comic scenes provide visual cues that help viewers perceive the sequence of comic panels as a fluid and ongoing action, making the scenes feel dynamic and alive. In addition, the chat system incorporates interactive elements such as chat bots and themed rooms to increase user engagement. In an extensive user study, we show that our novel system significantly enhances users' ability to convey emotions and engage in meaningful interactions online, and that users appreciate the wide range of self-expression options and personalised characters that allow for more nuanced communication. The versatility of generative AI approaches makes our platform suitable for a wide range of applications beyond mere chatting, including education, reducing language barriers and character-driven role-playing games.

1. Introduction

Communication on digital chat platforms loses many of the non-verbal aspects of interactions, such as tone of voice, expressions, and body language, resulting in conversations being prone to interpersonal misunderstandings. To bridge the gap between personal and remote exchanges, visual clues such as emoticons, are commonly used to clarify the tone of messages and convey nuances like irony. Subsequent advances in digital platforms have evolved these simple emoticons into more detailed graphics, such as yellow smileys, enriching the visual aspect of text-based interactions and ultimately leading to the development of emoji, which expand the standard set of smileys into thousands of small, text-sized graphics. In current chat systems, users also show great interest in personalizing messages with custom emoticons and a larger selection of sticker graphics that extend the selection of predefined smileys.

[1]Equal contribution.

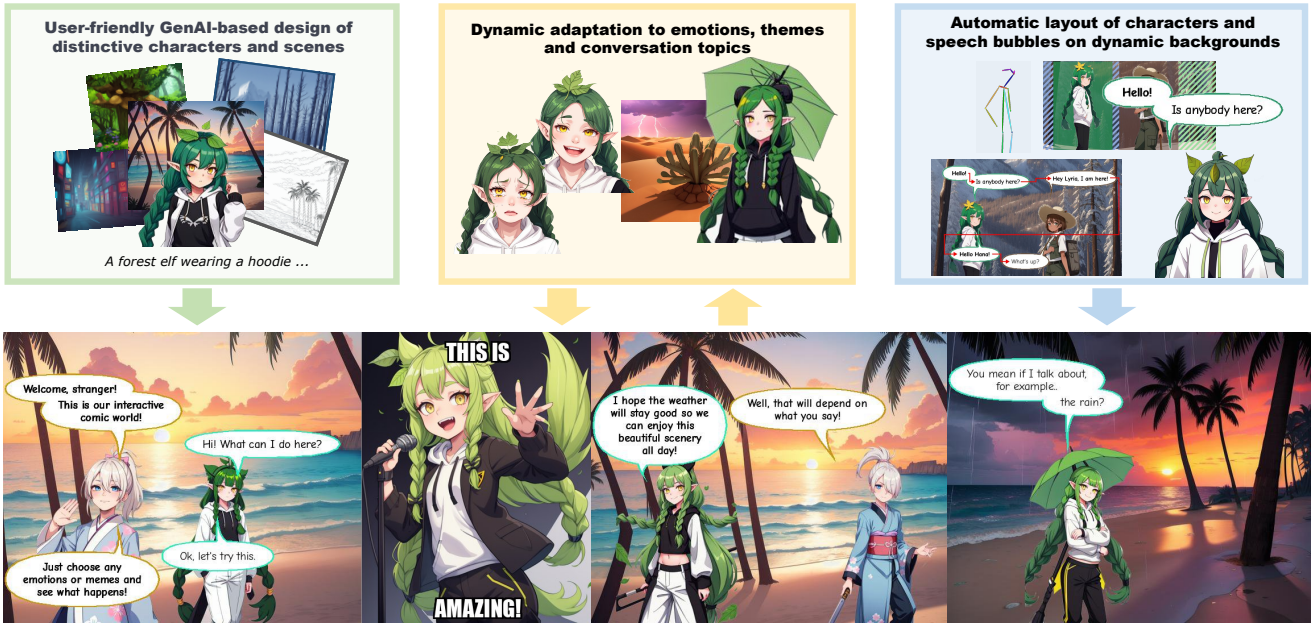

Fig. 1. Our interactive comic chat system integrates Gen-AI-based design tools, dynamically adapts characters and scenes to user input, and features a novel layout algorithm to generate panels.

The recent developments in deep generative models for image synthesis, especially with the open source release of Stable Diffusion[43], provide the ability to create appealing graphics from simple text prompts, opening the possibility of building powerful visual storytelling tools and communication systems that go beyond simple smiley graphics and standardized emoticons. In this paper, we explore how to use such models for communication by creating comics from text chats.

In 1996, Microsoft presented an interactive comic chat system [29] that automatically added visual representations to chat dialogue in the form of automatically generated comic panels depicting the people in the chat. Today, Microsoft Comic Chat looks outdated because its simple look does not fit well with today's expectations for high-resolution graphics, but the idea is as appealing today as it was in 1996, even if it needs a makeover. So we are building a new comic-based chat that takes advantage of modern technological advances, including generative AI, to give the idea a new spin.

Despite its success, a follow-up paper by Kurlander in 1998 [28] highlighted some *lessons learned* from the first version that remain relevant and are directly addressed by our new approach to the topic. One of the main sources of confusion in Microsoft Comic Chat was the small number of available characters, which led to multiple people using the same character in the same chat room, so one of the most requested features was the ability to draw your own characters. With our generative approach, user-created characters are one of the main features that make our chat interesting, as there are endless possibilities to design your own character using generative image AI. As part of our chat system, we developed a character design tool that is not constrained by a limited number of options, and by providing the option to create diverse characters, one has the means of self-expression, anonymity, and the possibility to present oneself differently in different contexts. It is known that the way people present themselves can have a profound effect on their self-perception and identification with the avatar and the group their avatar belongs to [40, 51], so we leave it up to the user to decide if they want to look similar or different from their usual appearance, or even create a completely new persona like a fantasy character. By allowing more than one character per user, users do not have to stick to one character idea, but have the flexibility to try out many different concepts and identities.

Another lesson learned from Microsoft Comic Chat was that people tend to *play act* the role of their character, so the design of the chat system influenced the kind of chats the users had. To build on this, our chat rooms offer a variety of (comic inspired) topics, illustrated by dynamically generated backgrounds that provide a consistent theme for the room. Characters and environments maintain their core characteristics throughout the chat, but adapt dynamically based on chat content and user input. These changes include expressing character emotions, changing poses and accessories, and external influences such as weather and lighting. These continuous changes provide visual cues that help viewers perceive the sequence of comic images as fluid and ongoing action, making the scenes appear lively and animated.

For backgrounds, using a generative image model means an infinite number of scene variations instead of a limited set of background images, and we can also adapt the scene shown in the background to the topics discussed in the chat. For example, an outdoor scene that normally depicts

a sunny day in the park can become rainy when users discuss bad weather. In addition, we can support different art styles in our scenes and provide a number of additional expressive features such as personalized memes. This opens up entirely new ways to quickly express and communicate individual emotions and themes visually. Last but not least, Microsoft Comic Chat modeled a very specific comic art style, including a relatively rigid bubble style, with only one bubble per character in each panel. It was optimized for small panels and for placing bubbles close together at the top of the panel. Our dynamic layout algorithm takes advantage of the larger screens available today and a variety of art styles with a more diverse panel composition. To do this, we design a more style-agnostic bubble style that focuses on readability and common modern comic panels.

Besides being entertaining, we believe that a comic chat can also help improve online communication. The many character personalization options allow people to express themselves, present themselves in the way they feel most comfortable, or try new things in a playful way. By using emotions expressed by the character, users can easily communicate their mood and the intended tone of conversations. We also believe that such a system could be helpful for neurodivergent people who may prefer online communication to face-to-face communication in some cases, but feel that text alone cannot communicate as much as face-to-face communication. In addition, images transcend language barriers and can, for example, make reading materials more accessible to non-native speakers.

Our main contributions are:

- A chat system based on our automatic comic generation that gives users a rich set of ways to visually express their emotions through dynamically adapted character images.
- A user-friendly Gen-AI-based design tool for personalized characters and scenes.
- A method for dynamically updating characters and scenes based on emotions, themes, memes, and conversation topics.
- A novel algorithm for optimal placement of characters and speech bubbles in dynamically updated comic panels, ensuring a clear reading order after each update.
- An evaluation of the system, including user perception of our comic-style representation of chat conversations.

2. Related work

People have been using the Internet as a medium for real-time text communication for decades, with IRC (Internet Relay Chat) [38] being one of the first popular chat systems, and many other text-based chats following, some using dedicated applications similar to IRC, and others implemented as web apps that run in the browser. The range of uses of chat systems is wide, from work communication over casual chatting to dating apps.

As an alternative to purely text-based chats, Microsoft released V-Chat in 1995, which allowed users to place an avatar on a background image showing a representation of the chat room, and in 1996 Microsoft Comic Chat [29, 28], an IRC client that automatically creates comic panels from a line-based text chat while remaining compatible with other clients that participate in the chat without comic visualization. The chat automatically creates panels, and users can manually select emotions that change the facial expression and pose of their comic character to convey the tone of the conversation. The messages are placed in speech bubbles, and the characters are placed so that they look at each other when in dialogue. Comic Chat was distributed until 1999 even though its usage declined over time. As another visual chat system, Itou et al. proposed a manga-style chat system in 2013 [25, 24], which allows people to choose different manga panel options, but is also limited to a combination of a predefined set of visual elements.

2.1. Automatic comic generation

Comic generation combines the development of visual art and sequential layout to create stories within a structured panel format. This process involves the controlled blending of text and images to communicate a variety of topics and emotions. Different works deal with the generation of artistic comic styles from a captured image of a scene [55] or from a drawing and a style reference using style transfer [31, 54, 57]. To bridge the gap in conveying emotions through text, [3] introduced a novel approach by generating speech bubbles that change shape based on the emotions of the speaker as predicted from voice recordings using Generative Adversarial Networks (GANs). In this context, [14] developed a technique for the automatic positioning of speech bubbles, a concept which also has been explored in [39] for captioning group conversations.

Since choosing the right comic layout is of great importance for storytelling and visual impact, and finding a good layout can be time-consuming, [8] proposed to automate the layout process in manga creation, and in their subsequent research [9] focused on generating the composition of subjects and speech bubbles. While these methods deal with the individual artistic elements and building blocks of comics, other works also address the creation of entire comic strips. Several approaches using different types of source materials have shown diverse applications for automatic comic generation. While [47, 20] proposed comics generated from source code, [12] transformed activity data into cartoon-style diaries, and [21, 53, 27, 13] converted movies into comics, demonstrating a variety of inputs that can be adapted into a comic form. To simplify the automated generation of comic strips, [1] proposed the *Comic Strip Description Language* (CSDL) to describe the elements and layout of comics.

2.2. AI chatbots for roleplaying

One of the first systems using language models for role playing was AIDungeon [37, 23] which started as a fine-tuned GPT-2 model [42] distributed as a Jupyter-Notebook, which later evolved into a role playing website [30] that does not require the users to run the language model themselves, but uses the OpenAI API on the server-side. With the advent of stronger open large language models (LLM) such as Llama [49, 50, 16], a new wave of AI roleplaying systems emerged, both hosted solutions that run models server-side and provide easy to use websites and mobile apps, and software projects like SillyTavern [15] for using LLMs for roleplaying on consumer hardware. These systems allow users to roleplay in a text chat, although they usually provide a profile image for the chatbot and sometimes a matching background image for the chat.

3. Overview

Figure 1 illustrates our interactive comic chat system which integrates Gen-AI based design tools and image generation, including conversation topics, emotions and other context to create lively comic panels. It features a novel layout algorithm for arranging characters and speech bubbles in dynamically updated comic panels. The subsequent sections are organized as follows: In section 4 we analyze the capabilities of diffusion models for automatically generating comic-style images, in section 5 we describe the character and scene design process and narrative image generation based on user-selected emotions and automatically detected topics, and in section 6 we describe how comic panels are dynamically generated from text input using our novel layout and speech bubble placement algorithms. After that, we evaluate our system and present a user study in section 7. Limitations and future work are discussed in section 8 before we conclude our work in section 9.

4. Generative AI for dynamic content

We leverage advances in generative AI, which has seen significant growth in recent years. These tasks include image generation, selective addition of new concepts, and controlling image generation by other means than text prompts. The techniques used are Stable Diffusion [43] as the image diffusion model, enhanced with low-rank adaptations [22] to add new styles and characters, and ControlNets [58] to enforce specific poses. We also incorporate large language models (LLMs) to generate prompts, classify image tags, and implement chatbots.

We use version 1.5 of Stable Diffusion as text-to-image model, of which fine-tuned versions are still popular and provide good quality at high speed, although larger models like Stable Diffusion XL (SDXL) are available and will be interesting for an future upgrade, possibly with extensions like SDXL Turbo [46] or SDXL Lightning [32]. There are a number of other new models that are interesting for future work (see sec. 8), but they are too slow for our purposes on current hardware.

The used diffusion model is conditioned by text inputs, called prompts in the rest of the paper, and the OpenPose [10] ControlNet. By using OpenPose, we always have the information about the location of the pose keypoints and can adjust them as needed. We also use other ControlNets, e.g., for conditioning with lineart, when fine-grained control over image generation is desired, and the Tile ControlNet to improve image quality when rendering large images. We utilize various LoRA models to add styles and characters, and to extend the base model with a Latent Consistency Model (LCM) [34]. As language model, we chose a fine-tuned Mistral 7B model [26], which proved to be a good compromise between quality and speed.

4.1. Stable Diffusion model

Fig. 2. Different Stable Diffusion models have significantly different styles. The characters in the image were rendered using the same prompt and seed with three different diffusion models.

For Stable Diffusion, there is a wealth of different models with different image styles available. Next to a few general-purpose models, many models specialize in a particular styles, such as photorealistic images, anime styles, semi-realistic images, or 3D images similar to computer games or 3D animated movies, as shown in Fig. 2. For our chat, we decided to use the "Cartunafied" model since we were looking for a comic aesthetic and because anime-style models are trained on tags rather than full-sentence captions, which makes it easier to create and (re)combine prompts.

Low-rank adaptations (LoRA)

Low-Rank Adaptation (LoRA) [22, 44] allows efficient fine-tuning of large pre-trained models by updating few parameters, allowing new content to be added to the base model without replacing existing content. The much smaller LoRA models can be dynamically added or removed, allowing the model to change styles and add new characters (see Fig. 3) with minimal computational overhead. In general, one can combine a base model with several different LoRA models during image generation, and decide which LoRA models to load on an image-by-image basis.

Fig. 3. Left: An image generated using the "Cartunafied" Stable Diffusion model with a LoRA model we trained on the Microsoft Comic Chat sprites for the character "Dan". Right: Two of the 27 original 155×224 sprites used to train the LoRA model.

4.2. Prompting anime models with Danbooru tags

In contrast to many other Stable Diffusion models, which are trained on datasets containing images from the web together with their captions or adjacent text, which usually follow a grammatical structure, anime models are mainly trained on the Danbooru dataset [2], which contains a large number of anime images with high-quality community-maintained metadata in the form of short tags assigned and maintained by members of the Danbooru image board site. These Danbooru tags describe the content of each image, as well as metadata about quality (e.g., `highres`) and issues such as incomplete tagging (e.g., `character request`). The community also links tags to their synonyms so that all tags are used consistently and each concept is represented by a unique tag from a known list of available tags.

All images in the Danbooru dataset have tags about the following categories: *meta*, *artist*, *copyrights* (e.g. the anime he characters are from), *character*, *general*. The general parts contain not only character-related tags like `blonde hair`, but also background details of the image that one would usually not think about directly, like `open door`, leading to captions that group images by things that can be seen in them. Diffusion models trained on these tags thus have the advantage that each training image was well-tagged, and that each concept has a unique keyword, so that training prompts are not fragmented by synonyms. The major advantage for our purposes is that the models were trained on comma-separated tags, which can be easily split and combined.

The overall structure for a character prompt to create an anime character thus usually starts with either `1boy` or `1girl`[2] and then lists visible properties such as hair color, clothing items, eye color, accessories, etc. as a comma-separated list of tags. The base prompt for a character can then be dynamically combined with further tags to change its appearance. For example, we allow users to choose which emotion to express by simply appending tags related to the emotion to the character prompt, as shown in Fig. 4. We also reinforce poses with tags like `solo, cowboy shot, standing`, or for face-only images, `face, portrait`, and to improve image segmentation, we add `simple background, grey background` to prompts for character sprites, see Fig. 10.

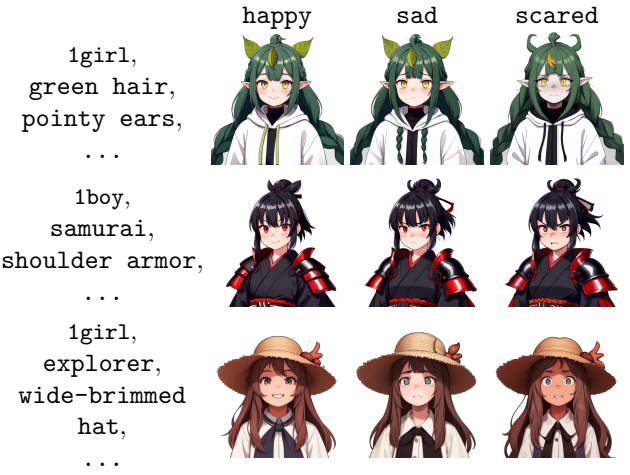

Fig. 4. We can change the expression of characters by appending tags to their base prompt. Our chat uses portrait images like these as personalized previews in the emotion selection dialog.

Image models also have a negative prompt to specify things that should not be in the image, where we use meta-tags like `bad anatomy` and `low quality`, which mark low quality images in the Danbooru dataset, to steer the diffusion model away from bad generations. We also use the negative prompt to enforce a negative bias towards undesirable image content such as nudity.

4.3. Considerations in image generation

In the context of unsupervised image generation for custom prompts, our approach acknowledges the inherent challenges posed by the absence of manual quality control. Typically, users creating high-quality AI images engage in multiple iterations, trying different seeds, using inpainting to rectify artifacts, and making adjustments in external graphics programs to achieve the desired result. Since our system has no way to automatically evaluate the quality of the generated images, it relies on the first generated image, which inevitably means that some images will exhibit visible artifacts. By choosing an anime model, we generate simpler graphics than photorealistic models, minimizing the impact of imperfections because the stylized nature of anime graphics is more forgiving of artifacts that would be more pronounced in realistic images. Moreover, we believe that users will value the ability to create personalized graphics over selecting from a limited set of pre-designed characters, even if they are not of the same quality as manually created images.

[2]Other or unclear genders are sometimes tagged as `1other`, but the tag does not work very well with image models

4.4. Image generation optimizations

To improve quality, we use features such as latent upsampling for higher resolution images, the OpenPose ControlNet to ensure the proper poses, and the Tile ControlNet when pre-rendering high-quality backgrounds.

To improve rendering speed, we use a Latent Consistency Model (LCM) [34] which allows us to reduce the number of diffusion steps from 20-30 to 4-7 at the cost of a slight loss in quality. In our experience, using LCM is feasible for dynamic backgrounds, while character sprites that can be reused can be worth the cost of rendering without LCM. We can easily switch at runtime between using it or not by using a LoRA model for the LCM [35].

5. Chat system

We use the automatic comic generation methods described in sec. 4 to build a web-based group chat in which users can chat using their self-designed graphical characters, in a variety of themed rooms. Fig. 5 shows an overview of the system and the following sections describe the steps for creating comic panels, which include creating character images, creating the panel backgrounds, and dynamically placing and moving the characters and speech bubbles in these panels as users write new chat lines.

5.1. Character design

The first step for new users is to create a character. The look of the character and if they want to present themselves or create a fantasy character is completely up to the users. To facilitate getting started, we offer a character creation page, that provides different options for defining characters:

- Create a random character.
- Create a character from a text description.
- Customize a pre-defined character template.
- Skip directly to the tag-based character editor.

The first two options are implemented by using an LLM to generate a tag-based image model prompt. In all four options, the user is then redirected to the character editor which provides a preview image and allows users to further customize the character to their liking. Screenshots of these pages are included in the supplemental material.

5.2. A user-friendly character editor

In the tag-based character editor, the user is presented with a character sheet like the ones artists use for concept art, and input fields for the image model prompts, in which the user can iteratively add and remove tags until the character sheet looks like what they imagined their character to look like. When the user is finished, the preview images used in the chat interface (see fig. 4) are generated, and the character is ready to join the chat. The Danbooru tags for character prompts are easy to combine in principle, but may not be intuitive for users who have never seen an example of a tagged image, so we worked with a user experience expert to create an interface that focuses on providing interactive lists of example tags for each section (e.g., hair, eyes, clothing) that can be added and removed from the prompt by clicking on them. This way, users can not only define basic characters using the example lists, but also get an idea of what the tags look like, so they can figure out tags that are not listed as examples. We also give a brief explanation of how to use tag weighting in prompts, how to use the negative prompt, and caveats such as contradictory tags.

Characters from descriptions

To create characters from descriptions, we use an LLM, so the user can write a simple description and our system generates the tags for the image model. A prompt like "A boy with short red hair and freckles wearing a blue coat and brown shorts" results in the tags `1boy, red hair, freckles, short hair, blue coat, brown shorts`, which follow the description verbatim. A description that only gives a basic idea and leaves the looks to the LLM, like "A mysterious superhero who rescues marine animals" also works and results in the tags `1girl`, `mask`, `cape`, `spandex suit`, `blue and white colors`, `animal motifs`, `flippers`, `diving suit`, `goggles`, `rescue equipment`, `strong and athletic build`, `blonde hair`. The avatars of the two example characters are shown in Fig. 6.

Personalization using LoRA models

Optionally, we offer users to use personalized avatars using LoRA models (see sec. 4.1) trained on their photos. Since LoRA models work well for style transfer, the resulting images still match the comic look of our chat, even though they were trained on photos. Currently, requests for personalized avatars are processed manually to ensure high quality results, but future work could consider automating the process using methods such as those proposed by Avrahami et al. [5]. Other ways to personalize characters are to compute text-encoder embeddings from reference images using textual inversion [18] or to use a reference image as an (additional) image prompt using IP-adapter [56].

5.3. Chat rooms

The chat is divided into several public and unlisted rooms and currently does not provide a private message feature. Chat rooms have a number of different scenes that are used when creating panels, which are defined by image model prompts that describe the backgrounds, and usually have a common theme related to the room. The room itself can also add details that are applied to all scenes of the room, such as a pixelation LoRA model for the video game room, and details that are applied to characters, such as adding `christmas hat` to the prompts of characters chatting in the Christmas room.

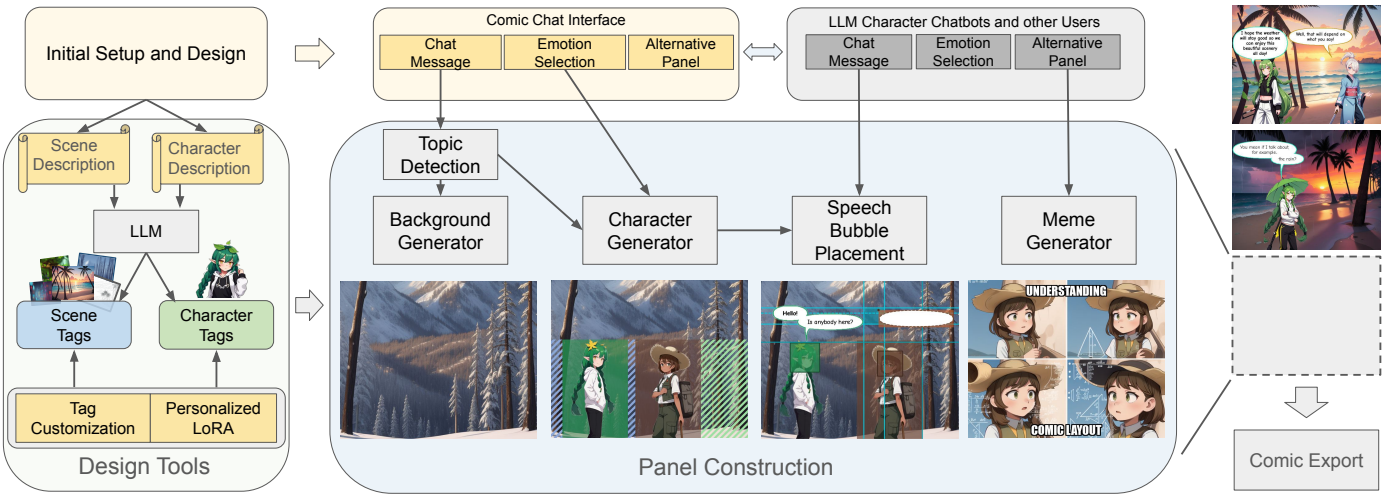

Fig. 5. Overview of how the different components of our comic chat system and design tools interact. Yellow fields mark user input.

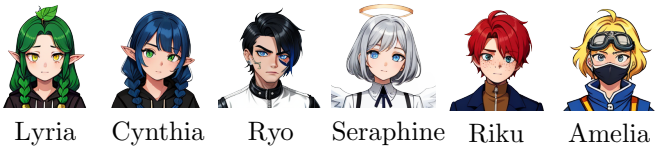

Fig. 6. Characters created using our character editor: Lyria (pre-made character), Cynthia (a user-edited version on Lyria), Ryo and Seraphine (random generated characters), Riku (the freckled boy described by his appearance) and Amelia (the superhero created using a role description).

Fig. 7. Rooms can have additional prompts that modify foreground and background. The "Monochrome nature" room uses the same scenes as the nature room (see Fig. 1), but adds a `monochrome` tag to scene and character prompts and the Christmas room adds the tag `christmas hat` to character prompts.

5.4. Using the chat

To guide new users through the features of the chat, upon registration users are directed to the character editor to create a first character for participating in the chat. Once a user has finished creating their character, a tutorial room is pre-selected on the main page that helps to introduce new users to the chat features in an interactive way. In the tutorial room, a chatbot guides the user step-by-step through the different features, waiting after each instruction for the user to try the function before explaining the next feature. The steps include, for example, selecting emotions and posting memes (see sec. 5.5), but leave it up to the user which emotion to select or which captions to use for the meme they need to create in order to progress through the tutorial.

While it is exciting for users to use personalized characters in rooms with dynamically generated backgrounds, we additionally leverage the possibility to adapt the graphics to the content of the chat. For emotions, we implement an emotion chooser similar to Microsoft Comic Chat, which allows the user to change the expression of their character, as well as detecting common emoticons like `:D`, `lol`, `>.<` and others, but we also detect conversation topics to adjust the background scene and the characters in the foreground. For example, if someone mentions *rain*, the background image changes to one with rain and the speaking character wears a raincoat or holds an umbrella, as shown in Fig. 1 and 10. Currently, we still use a manually maintained list of topics because detecting topics and creating consistent image model prompts from them is a challenging task. For example, one wants to avoid showing a happy background for a sad topic due to inaccurate topic detection, and some other conversation topics may be difficult to visualize. It would require a lot of tuning to ensure that an LLM can not only reliably extract topics, but also generate image model prompts that result in high quality images related to the topic.

5.5. Alternative formats

An alternative format to comic panels is posting Internet memes. The typical meme format contains a funny image, often a pop culture reference, with bold captions at the top and bottom of the image. In our chat, meme panels stand for themselves and do not contain speech bubbles. Instead of using the original meme image and just adding captions, as most meme generator sites do, we use a collection of LoRA models trained on various internet memes that allow us to reproduce the well-known image with the

Fig. 8. Left: Hana is in the math lady meme. Right: Lyria uses the Pepe Silvia meme.

person in the image replaced by the personalized character of the user. For some memes, an alternative to using LoRA models trained on the original images is to use ControlNets to combine, for example, a given line art sketch and a given pose to guide the image generation to an image that is recognizable to people who know the original meme. Fig. 8 shows examples in which the generated images contain our characters in two different meme formats that would normally use unmodified movie screenshots as backgrounds.

5.6. Integration of large language models

We implement several features that facilitate creation of characters and rooms using LLMs. For character creation, our system can generate image model tags from a free text description by prompting an LLM to extract tags for the visual properties of the described person, in the tag-based character editors for characters and room scenes, we provide a "suggest more tags" feature that uses the LLM to generate further tags to enhance the image model prompt, and we use the LLM to detect tags that might lead to not-safe-for-work images. Scenes can also contain a textual description instead of an image model prompt, which is then used to generate image model prompts using the LLM, so that we can generate a large number of different prompts from a single description of a general scene idea. Some rooms also feature chatbots, which are assigned a role by a short character description, allowing them to role-play their character traits with LLM-generated responses, emotions, and even memes. Finally, we summarize chats when exporting them as comics (as described in sec. 6.6) to automatically generate a catchy title that matches the content of the conversation.

6. Dynamic comic layout

In comic book design, the traditional approach to speech bubble placement requires that the panel layout be fixed before the speech bubbles are placed, including static positions for characters and their lines of dialogue, but these a priori methods are unsuitable in scenarios where future character interactions are unknown. To overcome this limitation, we propose a dynamic method for placing or updating each character and speech bubble immediately when the corresponding line of text is sent by the user. For the best visual flow and coherence, we place speech bubbles in a clear reading order with a dynamic speech bubble tail shape, close to the speaking character, without overlapping or obscuring important visual information of the panel image. In this way, our method allows the step-by-step construction of panels for new dialogue without major layout shifts, creating a composition that resembles typical comic panels. Because our layout algorithm is designed to be flexible, it easily adapts to different text directions and scripts, such as right-to-left scripts (e.g., for Arabic reading order) by reversing the order of vertical bubble placement and panel order. Similarly, our algorithm efficiently accommodates vertically oriented scripts, such as those commonly used in Japanese manga, by extending the speech bubbles vertically and changing the text wrapping to match this expectation.

6.1. Panel generation

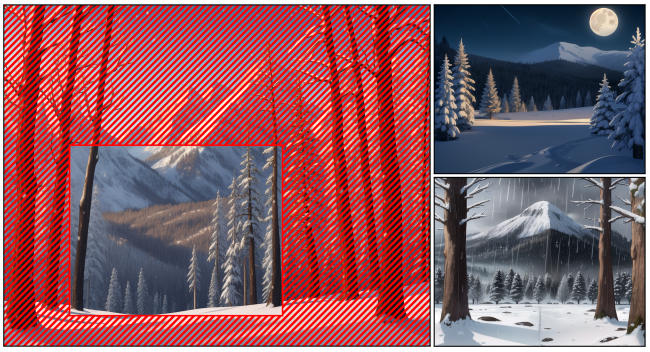

Fig. 9. Left: When we update a panel we alternate between different cutouts of a larger background image. Right: For some conversation topics we add related keywords like `night` or `rain` to the background prompt.

We create a new panel when a chat starts, or when either the speaker or the speech bubble for the new line does not fit into the current panel and otherwise we update the newest existing panel of the room with the new chat line.

To create a new panel, we select a random scene prompt from the scenes of the room and use it to draw the background image. For normal panels, we render the background at twice the panel size, cache the rendered image, and each time the panel is updated, we scale the image down by a random factor between 1 and 2 and then create a random cutout to make it look like the characters are moving around the scene, similar to [29], see Fig. 9 and the bottom row in Fig. 1. For panels influenced by a topic, we currently always generate new images at panel size and do not use our cutout approach to make these panels more dynamic. After we have created a new panel or when we update an existing one, we use algorithm 1 to place the speaker in the panel.

6.2. Character rendering

To render a character, we first assemble its prompt from the base prompt, the chosen emotion and accessories

as shown in Fig. 4, and possibly an additional prompt defined by the room (cf. Fig. 7). Next, we need to figure out the viewing direction of the character in order to select a pose using the following heuristic:

- If the character is the first character in the panel and is addressing another character, we use a right-facing pose because the next character will be placed to the right.
- If the character is not yet in the panel but addressing a character in the panel we use a left-facing pose, because it will be placed to the right of the addressed character.
- If the character is already in the panel and talking to another character who is already in the panel, we have it face the other character, determining whether it is left-facing or right-facing by the order of the characters in the panel.
- Otherwise, we let the character look at the viewer.

A future direction is to incorporate more of the conversation flow into this heuristic when no character is directly addressed, e.g. using the LLM to analyze who is talking to whom.

Fig. 10. We use an OpenPose skeleton to reinforce poses using a ControlNet and render the characters with the additional tags `simple background`, `grey background` which reliably yield a static grey background that works well for background segmentation using ISNet. The last image shows an image that uses the same pose and seed, but has the tag `rain coat` added to the prompt because of a weather conversation topic.

Once we know the viewing direction, we use the prompt and a random pose for the chosen viewing direction to create a character image, and remove the image background using ISNet [41] as shown in Fig. 10. To avoid placing speech bubbles over the faces of the characters, we also store the face position, which is calculated from the bounding box of the OpenPose face keypoints of the selected pose, shown as white box in the skeleton image. We cache the generated character sprites to speed up later generations that use the same pose, but use a set of multiple poses per viewing direction to vary the sprites in each panel. With different skeletons, viewing directions, emotions, and additional changes based on conversation topics, we generate a large number of different character images.

6.3. Character placement

We assume a fixed width for our characters, regardless of the transparency mask of the actual character, which is the same for all characters, because we always use the `cowboy shot` perspective (as shown in Fig. 2 and 10) for characters in chat panels. This not only makes it easier to avoid overlaps, but also ensures that the characters are not packed too tightly into the panel.

Fig. 11. New characters are always placed in the green area to the right. If there is not enough space for a new character, other characters are moved to the left, reducing the space between them (marked in blue).

Algorithm 1 then updates the panel as follows. New characters are always placed to the right to extend the group of characters without disrupting the previous order of characters, and as an easy way to ensure that a character that can be placed in the panel can also have a speech bubble placed in the correct reading order. If a panel has enough space between the rightmost character and the right panel boundary, we place the character at a random position within that area, maintaining the position of the other characters. If the area is too small, but the total free space in the panel would fit another character (see Fig. 11), we move the other characters to the left to make room for the new character using algorithm 2. If the new character does not fit into the existing panel, we create a new panel and can then place the character there.

6.4. Speech bubble placement

After the characters are placed in the panel, we place the speech bubbles near the heads of the corresponding characters, with the tail pointing toward the mouth, as determined by the face bounding box using the following algorithm.

When we have already placed a set of bubbles $\mathcal{B} = \{B_1, \ldots, B_n\}$, each associated with a character $c(B_i)$ within the panel, we need to find a free space for the new bubble B_{n+1}, and since the bubbles are created dynamically whenever a user writes a new line, the total number is not known in advance. Since we only know the position of $B_1, \ldots, B_n$ at the time bubble B_{n+1} is placed, and cannot predict which positions would be suitable for further bubbles of yet unknown text length, our algorithm can only

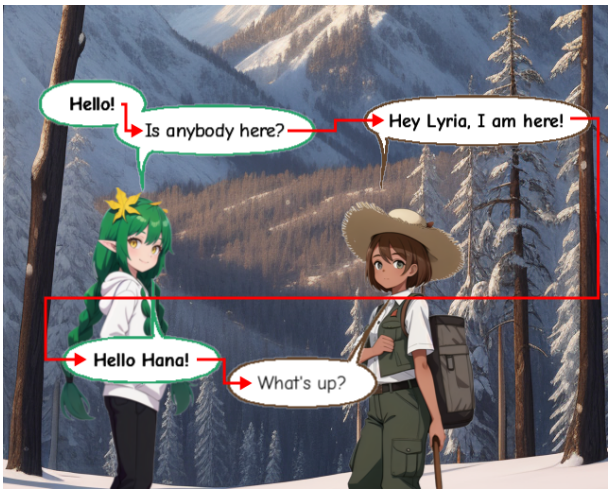

Fig. 12. The bubble layout must follow an intuitive reading order to avoid misunderstandings.

heuristically place bubbles in a way that leaves room for more bubbles. The bubble placement also must meet the following constraints:

- B_{n+1} should be placed near the face of the corresponding character to create a visual association between the bubble and the speaker and to facilitate tail placement.
- The bubble layout should follow the left-to-right, top-to-bottom reading order of the English language as shown in Fig. 12.
- Bubbles must not intersect with character heads, speech bubbles of other characters, or the boundary of the panel.
- Bubbles belonging to the same speaker may overlap as long as the text is clearly separated.

Although the reading order in principle forbids placing bubble B_{n+1} with its top edge higher than B_n (dashed line in Fig. 13), we found that this constraint can be relaxed based on the horizontal separation of the bubbles, so we extend the admissible upper limit for B_{n+1} by a third of the height of B_n to avoid a rigid, stair-like appearance.

To place a new bubble, we first determine the size of the bubble bounding box by calculating the text bounding box and adding the padding needed to enclose the text box with a bubble. For a balanced bubble size, we calculate the text width using the algorithm from [29] and center the lines inside the bounding box. Next, we calculate the constraints on where the bubble bounding box can be placed without intersecting other image elements. To do this, we calculate the admissible area in which the upper-left vertex of the bounding box can be placed (shown in Fig. 13) using algorithm 3. The left side is defined by the right edge of the bounding box of the previous bubble, the right side is the right edge of the speaker's character sprite, the bottom is defined by the uppermost face bounding box, and the top is defined by the top of the previous bubble plus one third of the height of the previous bubble. When placing the bubble, we further reduce the limits so that the bubble cannot intersect the panel boundaries.

We place subsequent bubbles created for the same speaker close to the previous bubble, allowing some overlap, defined by a fixed vertical distance d_v that ensures the text does not overlap but is clearly recognizable as a continuation of the previous text, and a minimum horizontal distance d_h that ensures a continuous reading order, see Fig. 14. Such merged bubbles are common in comics and make the conversation look more coherent when the same person is speaking twice. For such groups of bubbles, only one tail is drawn, connected to the bubble closest to the speaker.

If a bubble cannot be placed in the top row due to these constraints, we continue by placing bubbles below the character faces using a similar algorithm, except that we then use the lowest face bounding box as the upper limit for bubble placement. In particular, if we add a bubble for a character to the left of the previous speaker, we must always change the row to maintain the correct reading order. If a bubble for a long text does not fit in the available space, we try to split it into a smaller bubble that can be placed, and create additional bubbles for the rest of the text. This is especially relevant for bubbles that would not even fit into a new panel, and could not be placed at all without breaking them up.

Fig. 13. The current bubbles and head bounding boxes determine the free space where algorithm 3 places new bubbles. The dashed line shows the upper limit for non-increasing placement, while the solid lines show the limits of the slightly relaxed requirement. The upper left corner of the bounding box must be inside the blue shaded area. Note that the origin is in the upper left corner of the panel.

6.5. Drawing speech bubbles

We modeled our speech bubble design after modern comics, but kept it as style agnostic as possible to allow for the variety of art styles that image models can accommodate. We focus on left-to-right reading order and generate bubbles with the major axis of the roughly elliptical shape

in the horizontal direction. To reproduce a typical bubble shape, we use Bézier curves to create an elliptical shape around the text. The outline consists of four quadratic Bézier curves defined by a rectangle around the text with padding to avoid text outside the shape. The start and end points of the curves are placed at the midpoints of each edge, and the control points are placed at the corners of the bounding box.

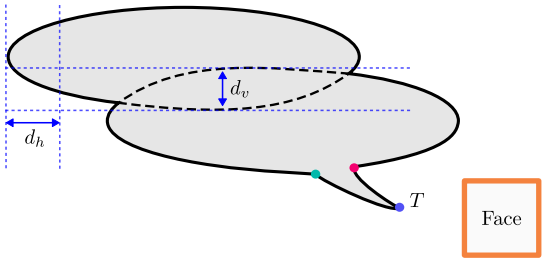

Fig. 14. When a character speaks twice, we combine the bubbles while still maintaining a clear reading order and preventing text overlap. A tail is only attached to the bubble closest to the speaker.

The bubble tail is drawn using two Bézier curves, each connecting a point on the bubble boundary to the tail tip T near the speaker's face, forming a curved tail pointing toward the face of the speaker, as shown in Fig. 14. Whether the tail points inward or outward depends on the relative position of the speaker, and the starting position is randomly chosen on the side of the bubble closer to the speaker to ensure a clear visual connection. The width of the tail depends on the ratio of the length of the tail to the width of the bubble, which has proven to have a aesthetic look. The curvature of the Bézier curves is calculated based on the angle to the speaker and the slope between the curve start points on the bubble.

6.6. Comic strip export

Fig. 15. Users can download their chats formatted as typical comic book pages with an automatically generated title.

For many purposes, it is beneficial not only to chat interactively, but also to be able to download the resulting comic, allowing users to revisit the discussion, preserve the insights of the conversation, and enjoy the humor in the panels. In addition to a download button for individual panels, we provide an export function that creates comic pages from the conversation by arranging the panels in a grid, splitting longer conversations into multiple pages, and using the LLM to generate a catchy title from the chat show on the page. Figure 15 shows an example comic strip with three panels.

7. Evaluation

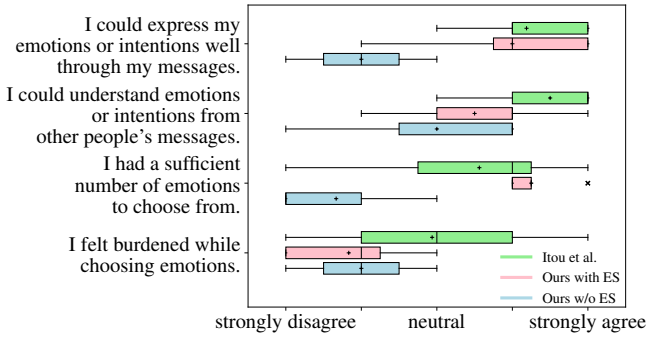

Fig. 16. Comparison of the emotional expressiveness in our comic chat and the manga chat system by [24]

To evaluate our system in terms of usability, emotional expressiveness, effectiveness of text-based design tools, and panel generation speed, we made a prototype of our chat available in closed alpha and beta testing phases. During the alpha tests, our users were free to choose how they wanted to use the system, while we asked them directly for feedback on how to improve the chat, so that we could understand the problems and technical drawbacks encountered, as well as the needs and desires of our users. During our beta testing phase, we invited external users to participate in a closed beta test. They were asked to fill out a survey after using the system to create a user study for quantitative evaluation.

The 15 participants in our user study were asked to create their own character using our design tool, followed by an optional tutorial explaining all the features of the chat, and then to chat in one of 14 different themed rooms we provided, three of which included a chatbot. The survey consisted of questions to understand the general chat usage history of our participants, questions about the usability of our systems, and a third segment asking detailed questions about our various features. All of our participants were familiar with and regularly used common chat systems.

7.1. User interface design

To evaluate our current user interface, we computed the System Usability Score (SUS) [6], which consists of a set of ten questions rated on a five-point Likert scale, where a score of 1 indicates strong disagreement with a statement and a score of 5 indicates strong agreement. A score from 0 to 100 is calculated from the responses, with 100 indicating high subjective satisfaction with the usability of the system. The resulting score of 65.3 indicates that the interface still needs improvement, which may be partly due to the fact that a group of users did not take

the tutorial – as a result, they were not aware of the Emotion Selection (ES) feature. Figure 16 shows how omitting these users from the test evaluation gives a clearer picture of overall user satisfaction, as most other users found the chat fun and easy to use. It does, however, highlight the importance of designing a user interface that allows users to intuitively find and understand all the features of the system. Another critical issue that the survey revealed was disapproval of image loading times (see 7.4), especially when a large group of people were generating images at the same time. Longer loading times tended to cause users to post several short messages in a row, which in turn tended to branch the conversation into multiple topic threads within the same chat room, causing confusion.

7.2. Emotional expressiveness

The survey questions about our various chat features, such as emotion selection and meme creation, were also rated on a five-point Likert scale. On average, users felt that their characters were relatable to them and that choosing emotions for their personal characters helped them express their feelingsand found the number of available emotions sufficient. This result suggests that visualizing emotions in personally designed characters helps people communicate their feelings. The personalized memes (sec.5.5) were also found to be helpful, and users mostly agreed that the images were visually appealing and added excitement to the conversation. Users also found that the speech bubbles were clearly associated with the speaker (sec. 6.4).

In addition, we compared our chat system with the manga-style chat system of [24], which is the closest comparable system since Microsoft Comic Chat, in terms of emotional expressiveness, and similar to us, they conducted a study with 12 users. Fig. 16 shows that users of both systems felt they were able to express their emotions reasonably well, while their users also seemed to find it easier to understand the emotions of their chat partner. So while understanding emotions or intentions was generally rated slightly higher in the manga chat system, this is partly because the waiting times for messages (see 7.4) and a larger group of people chatting at the same time caused some confusion in our test setup, as some users noted, and [24] only tested their chat with pairs of users. Also, they compared their system to a text-only chat system, which may have led to a more favorable result in terms of comparative expressiveness. Importantly, both groups felt they had a sufficient number of emotions to choose from, but our users were significantly less overwhelmed when choosing emotions. This demonstrates the usability of our emotion selection interface, which displays personalized previews (see Fig. 4).

7.3. Character design

Users were asked to rate the character design tool described in sec. 5.2. The preference for writing a description versus using tags directly for character design was inconclusive – further investigation of this question would require a larger sample size and a clear distinction between users working only with a text description and users refining character prompts with tags.

The ease of designing a character was rated with an average score of 3.2 – the tag customization editor was significantly more complex than our default option of providing simple text input, including detailed explanations of tag selection and weighting - this may have led users working with the complex part of the editor to rate the system as more difficult to use. We chose to allow editing the raw prompt for customization, prioritizing good personalization options over simplicity of the interface. In the future, it would probably be useful to distinguish between users who want a simple interface and users who want the most flexibility in creating their character, and provide them with different interfaces.

7.4. Panel generation time

The image generation on our test system with a single NVIDIA 4090 graphics card took about 2.8 seconds to create a character sprite including background segmentation (see sec. 6.2) at 512×768, about 4 seconds to create a new background at 1280×1024 (640×512 with latent upscaling), and about 1 second to create a background at 640×512 (see sec. 6.1). The processing time for creating a background prompt using the LLM was about 0.5 seconds, and generating bot responses took an average of 0.6 seconds. We did not measure the time for image compositing and speech bubble generation (using CPU), but it was not a limiting factor compared to image generation.

8. Limitations and future work

Our work aims to provide a real-time comic generation system. This comes with certain trade-offs, such as fast image generation being more important in our application than having the highest quality images, as we have seen in our beta tests. Similarly, we use a simplified panel layout where each panel is the same size for simplicity and to be able to wrap the panels according to the user's screen size, which is especially important for mobile devices. Another limitation is that we cannot easily detect unrealistic height differences between characters and background objects in the generation scenes. Backgrounds with landscapes and large places like the interior of a cave worked fine, but scenes in e.g. office rooms resulted in characters being disproportionately large or small compared to the furniture, and our approach of randomly zooming into the background cutouts (see sec. 6.1) cannot be used for such scenes.

Furthermore, since we are using generative text and image generation models, the typical problems of these methods are also relevant to our system. Not all generated

images and text will match user expectations, and topic detection, for example, could produce semantically inconsistent image model prompts that perform worse than handcrafted prompts. In addition, there is a risk that images may not always be "safe for work" and even seemingly harmless prompts could potentially trigger unwanted content. Our system strives to give users as much creative freedom as possible, but we recognize that this freedom can be abused, so we filter certain tags and use negative prompts for the image generation model. Should inappropriate content become an issue, we will consider using an image safety checker before adding character images to panels.

Computational cost

The image generation with Stable Diffusion is the computational bottleneck, compared to which the remaining processing time is negligible. Since renting servers with graphics cards capable of running neural nets is more expensive than renting regular servers, an important point in scaling the system will be to use the GPUs efficiently so that images can be delivered in time, but there is little time when the GPUs are idle but still paid for. One option to explore is to use an image generation API provider that charges per image or by processing time and does not charge if no image is generated, rather than renting servers for image generation.

Future models

We will evaluate the use of upcoming Stable Diffusion models, such as Stable Diffusion 3 [17], even though current results using the Stable Diffusion 3 base model are inferior to those of its predecessors in many use cases, and lack of quality and licensing issues have reduced the interest of the Stable Diffusion community in creating fine-tuned models and LoRA models. For Stable Diffusion 1.5 and SDXL, another interesting new direction may be to replace the Latent Consistency Model (LCM) with a Phased Consistency Model (PCM) [52].

While the quality issues of Stable Diffusion 3 may be resolved if the community adopts the model, we will also evaluate other completely different text-to-image models, such as Kandinsky [4], Lumina-T2X [19], PixArt-Σ[11], Kolors [48], AuraFlow [45], Flux [7], and Lumina-mGPT [33]. There are also many new language models that could be good alternatives to Mistral 7B. Two notable releases worth considering for systems like ours are Llama3 8B [16] and Mistral-Nemo 12B [36], both of which are quite powerful models despite having a small number of parameters.

9. Conclusion

In this paper, we have effectively used advanced large-scale Gen-AI models to develop an innovative comic chat system that automates the generation of comics from conversations. Our system incorporates Gen-AI-based design tools that allow for extensive character and scene personalization, and features a layout algorithm that dynamically arranges characters and speech bubbles to ensure clear reading order and coherent visual storytelling as new chat lines are introduced. Furthermore, it dynamically adapts background images and characters to conversation topics and user input, including emotions and other contextual elements, providing additional visual cues that allow viewers to perceive the sequence of comic panels as fluid and dynamic. Our evaluation, based on user studies, shows that personalized characters and the wide range of expression options significantly improved user interaction and the ability to communicate emotions and ideas online. Users appreciated the ease of conveying emotions and found the comic-style format of chat conversations engaging and intuitive. By combining the power of visual storytelling with the interactivity of digital chat, our comic chat system not only makes conversations more engaging, but also bridges the gap between text-based and visual communication, providing an effective way to convey emotions and ideas in digital conversations.

References

[1] Tiago Alves, Adrian McMichael, Ana Simões, Marco Vala, Ana Paiva, and Ruth Aylett. Comics2d: Describing and creating comics from story-based applications with autonomous characters. *Proceedings of CASA*, 2007.

[2] Anonymous, Danbooru Community, and Gwern Branwen. Danbooru2021: A large-scale crowdsourced and tagged anime illustration dataset., 2022.

[3] Toshiki Aoki, Rintaro Chujo, Katsufumi Matsui, Saemi Choi, and Ari Hautasaari. Emoballoon-conveying emotional arousal in text chats with speech balloons. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–16, 2022. https://dl.acm.org/doi/pdf/10.1145/3491102.3501920.

[4] Vladimir Arkhipkin, Andrei Filatov, Viacheslav Vasilev, Anastasia Maltseva, Said Azizov, Igor Pavlov, Julia Agafonova, Andrey Kuznetsov, and Denis Dimitrov. Kandinsky 3.0 technical report, 2023.

[5] Omri Avrahami, Amir Hertz, Yael Vinker, Moab Arar, Shlomi Fruchter, Ohad Fried, Daniel Cohen-Or, and Dani Lischinski. The chosen one: Consistent characters in text-to-image diffusion models. *arXiv preprint arXiv:2311.10093*, 2023.

[6] Aaron Bangor, Philip T Kortum, and James T Miller. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[7] Black Forerst Labs. Announcing black forest labs, August 2024.

[8] Ying Cao, Antoni B Chan, and Rynson WH Lau. Automatic stylistic manga layout. *ACM Transactions on Graphics (TOG)*, 31(6):1–10, 2012.

[9] Ying Cao, Rynson WH Lau, and Antoni B Chan. Look over here: Attention-directing composition of manga elements. *ACM Transactions on Graphics (TOG)*, 33(4):1–11, 2014.

[10] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[11] Junsong Chen, Chongjian Ge, Enze Xie, Yue Wu, Lewei Yao, Xiaozhe Ren, Zhongdao Wang, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-σ: Weak-to-strong training of diffusion transformer for 4k text-to-image generation, 2024.

[12] Sung-Bae Cho, Kyung-Joong Kim, and Keum-Sung Hwang. Generating cartoon-style summary of daily life with multimedia mobile devices. In *New Trends in Applied Artificial Intelligence:*

20th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2007, Kyoto, Japan, June 26-29, 2007. Proceedings 20, pages 135–144. Springer, 2007.

[13] Wei-Ta Chu, Chia-Hsiang Yu, and Hsin-Han Wang. Optimized comics-based storytelling for temporal image sequences. *IEEE transactions on multimedia*, 17(2):201–215, 2014.

[14] Bong-Kyung Chun, Dong-Sung Ryu, Won-Il Hwang, and Hwan-Gue Cho. An automated procedure for word balloon placement in cinema comics. In *Advances in Visual Computing: Second International Symposium, ISVC 2006 Lake Tahoe, NV, USA, November 6-8, 2006. Proceedings, Part II 2*, pages 576–585. Springer, 2006.

[15] Cohee, RossAscends, and the SillyTavern community. SillyTavern: LLM Frontend for Power Users, 2024.

[16] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407, 2024.

[17] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. *arXiv e-prints*, pages arXiv–2403, 2024.

[18] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion, 2022.

[19] Peng Gao, Le Zhuo, Dongyang Liu, Ruoyi Du, Xu Luo, Longtian Qiu, Yuhang Zhang, Chen Lin, Rongjie Huang, Shijie Geng, Renrui Zhang, Junlin Xi, Wenqi Shao, Zhengkai Jiang, Tianshuo Yang, Weicai Ye, He Tong, Jingwen He, Yu Qiao, and Hongsheng Li. Lumina-t2x: Transforming text into any modality, resolution, and duration via flow-based large diffusion transformers, 2024.

[20] David Heidrich and Andreas Schreiber. Visualizing source code as comics using generative ai. In *2023 IEEE Working Conference on Software Visualization (VISSOFT)*, pages 40–44. IEEE, 2023.

[21] Richang Hong, Xiao-Tong Yuan, Mengdi Xu, Meng Wang, Shuicheng Yan, and Tat-Seng Chua. Movie2comics: a feast of multimedia artwork. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 611–614, 2010.

[22] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[23] Minh Hua and Rita Raley. Playing with unicorns: Ai dungeon and citizen nlp. *DHQ: Digital Humanities Quarterly*, 14(4), 2020.

[24] Junko Itou, Kazuho Matsumura, Jun Munemori, and Noboru Babaguchi. A comic-style chat system with japanese expression techniques for more expressive communication. In *Collaboration Technologies and Social Computing: 25th International Conference, CRIWG+ CollabTech 2019, Kyoto, Japan, September 4–6, 2019, Proceedings 25*, pages 172–187. Springer, 2019.

[25] Junko Itou, Yuichi Motojin, and Jun Munemori. Development of manga-style chat system aiming to communicate nonverbal expression. *Procedia Computer Science*, 22:745–752, 2013.

[26] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[27] Guangmei Jing, Yongtao Hu, Yanwen Guo, Yizhou Yu, and Wenping Wang. Content-aware video2comics with manga-style layout. *IEEE Transactions on Multimedia*, 17(12):2122–2133, 2015.

[28] David Kurlander. Comic chat: From research to product. *Interaction'98, Mar*, pages 1–4, 1998.

[29] David Kurlander, Tim Skelly, and David Salesin. Comic chat. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 225–236, 1996.

[30] Latitude Games. AI Dungeon (Website), 2024.

[31] Junjian Lian and Jinrong Cui. Anime style transfer with spatially-adaptive normalization. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2021.

[32] Shanchuan Lin, Anran Wang, and Xiao Yang. Sdxl-lightning: Progressive adversarial diffusion distillation, 2024.

[33] Dongyang Liu, Shitian Zhao, Le Zhuo, Weifeng Lin, Yu Qiao, Hongsheng Li, and Peng Gao. Lumina-mgpt: Illuminate flexible photorealistic text-to-image generation with multimodal generative pretraining, 2024.

[34] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference, 2023.

[35] Simian Luo, Yiqin Tan, Suraj Patil, Daniel Gu, Patrick von Platen, Apolinário Passos, Longbo Huang, Jian Li, and Hang Zhao. Lcm-lora: A universal stable-diffusion acceleration module. *arXiv preprint arXiv:2311.05556*, 2023.

[36] Mistral AI team. Mistral nemo. *Mistral Blog*, 2024.

[37] Nick Walton Alan Walton and Max Robinson. AIDungeon (GPT-2 Code), 2019.

[38] J. Oikarinen and D. Reed. Internet Relay Chat Protocol. RFC 1459, May 1993.

[39] Yi-Hao Peng, Ming-Wei Hsi, Paul Taele, Ting-Yu Lin, Po-En Lai, Leon Hsu, Tzu-chuan Chen, Te-Yen Wu, Yu-An Chen, Hsien-Hui Tang, et al. Speechbubbles: Enhancing captioning experiences for deaf and hard-of-hearing people in group conversations. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–10, 2018.

[40] Anna Samira Praetorius and Daniel Görlich. How avatars influence user behavior: A review on the proteus effect in virtual environments and video games. In *Proceedings of the 15th International Conference on the Foundations of Digital Games*, pages 1–9, 2020.

[41] Xuebin Qin, Hang Dai, Xiaobin Hu, Deng-Ping Fan, Ling Shao, and Luc Van Gool. Highly accurate dichotomous image segmentation. In *ECCV*, 2022.

[42] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[43] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[44] Simo Ryu. Low-rank Adaptation for Fast Text-to-Image Diffusion Fine-tuning, 2023.

[45] Simo Ryu and fal. Introducing AuraFlow v0.1, an Open Exploration of Large Rectified Flow Models, July 2024.

[46] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation, 2023.

[47] Sangho Suh, Jian Zhao, and Edith Law. Codetoon: Story ideation, auto comic generation, and structure mapping for code-driven storytelling. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, pages 1–16, 2022.

[48] Kolors Team. Kolors: Effective training of diffusion model for photorealistic text-to-image synthesis. *arXiv preprint*, 2024.

[49] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[50] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[51] Jan Van Looy, Cédric Courtois, and Melanie De Vocht. Player identification in online games: Validation of a scale for measuring identification in mmorpgs. In *Proceedings of the 3rd International Conference on Fun and Games*, pages 126–134, 2010.

[52] Fu-Yun Wang, Zhaoyang Huang, Alexander William Bergman,

Dazhong Shen, Peng Gao, Michael Lingelbach, Keqiang Sun, Weikang Bian, Guanglu Song, Yu Liu, et al. Phased consistency model. *arXiv preprint arXiv:2405.18407*, 2024.

[53] Meng Wang, Richang Hong, Xiao-Tong Yuan, Shuicheng Yan, and Tat-Seng Chua. Movie2comics: Towards a lively video content presentation. *IEEE Transactions on Multimedia*, 14(3):858–870, 2012.

[54] Der-Lor Way, Wei-Cheng Chang, and Zen-Chung Shih. Deep learning for anime style transfer. In *Proceedings of the 2019 3rd international conference on advances in image processing*, pages 139–143, 2019.

[55] Tongtong Wei and Lianxiang Zhu. Comic style transfer based on generative confrontation network. In *2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP)*, pages 1011–1014. IEEE, 2021.

[56] Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models, 2023.

[57] Lvmin Zhang, Yi Ji, Xin Lin, and Chunping Liu. Style transfer for anime sketches with enhanced residual u-net and auxiliary classifier gan. In *2017 4th IAPR Asian conference on pattern recognition (ACPR)*, pages 506–511. IEEE, 2017.

[58] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3813–3824. IEEE, 2023.

Appendix A. Placement algorithms

We show the pseudocode for our simple but effective algorithms for placing characters and speech bubbles in chat panels. Algorithm 1 updates a panel for a new chat line and places the speaking character in the panel if it is not already there. Algorithm 2 moves the existing characters to make room for an additional character in the panel, and algorithm 3 finally places the bubble for the new chat line.

Algorithm 1: Updating a panel for a new chat line

Data: Panel P with n characters $c_1, \ldots, c_n$
Input: Speaker c, text t
Output: Modified or new panel P
if not $c \in P$ **then**
  `// Add the character to the panel`
  if $n < n_{max}$ **then**
    if $P.width - c_n.right < c.width$ **then**
      move_characters(P, c)
    end
    left $\leftarrow$ random($c_n.right, P.width - c.width$)
  else
    P $\leftarrow$ new_panel()
    left $\leftarrow$ random($0, P.width - c.width$)
  end
  create_sprite(c);
  place_character($P, c, left, P.height - c.height$)
else
  `// Generate the new emotion, pose, etc.`
  update_sprite(c)
end
$B \leftarrow$ create_bubble(t) place_bubble(P, c, B)
draw_tail(B, $c.face$);
return P

Algorithm 2: Move characters to make room for a new one

Data: Panel P with characters $c_1, \ldots, c_n$
Input: New character c_{n+1}
Output: Modified panel P
$left \leftarrow 0$ `// Leftmost admissible position`
for $i \in \{1, \ldots, n\}$ **do**
  `// Leave space for` $n - i$ `characters`
  $c_i.left \leftarrow$
  $random(left, P.right - \sum_{j=i+1}^{n} width(c_j))$
  $left = c_i.left + c_i.width$
end
return P

Algorithm 3: Placing a bubble in the top row of a panel

Data: Current Panel P with characters $c_1, \ldots, c_m$, upper row bubbles $B_1, \ldots, B_n$
Input: Speaker $c_i \in \{c_1, \ldots, c_m\}$, New bubble B_{n+1}
Output: Modified or new panel P

```
/* Check correct reading order for
   character c_i and character c(B_n)        */
```

if $c_i.left > \mathrm{c}(B_n).left$ **then**
  $box\_left \leftarrow B_n.right$
  $box\_top \leftarrow \max(0, B_n.top - B_n.height/3)$
  $box\_right \leftarrow$
   $\min(\mathrm{c}(B_n).right, P.width - B_{n+1}.width)$
  $faces\_top \leftarrow \min_j(c_j.face\_top)$
  $box\_bottom \leftarrow \max(0, faces\_top - B_{n+1}.height)$
  if $box\_left \leq box\_right$ **and**
   $box\_top \leq box\_bottom$ **then**
    $B_{n+1}.left \leftarrow \mathrm{random}(box\_left, box\_right)$
    $B_{n+1}.top \leftarrow \mathrm{random}(box\_top, box\_bottom)$
  else
    $P \leftarrow \mathrm{place\_in\_bottom\_row}(c_i, B_{n+1})$
  end
else
  $P \leftarrow \mathrm{place\_in\_bottom\_row}(c_i, B_{n+1})$
end
return P