Comic Chat

David Kurlander
Microsoft Research
One Microsoft Way
Redmond, WA 98052
djk@microsoft.com

Tim Skelly
Microsoft Research
One Microsoft Way
Redmond, WA 98052
timsk@microsoft.com

David Salesin
Department of Computer Science
University of Washington
Seattle, WA 98195
salesin@cs.washington.edu

ABSTRACT

Comics have a rich visual vocabulary, and people find them appealing. They are also an effective form of communication. We have built a system, called Comic Chat, that represents on-line communications in the form of comics. Comic Chat automates numerous aspects of comics generation, including balloon construction and layout, the placement and orientation of comic characters, the default selection of character gestures and expressions, the incorporation of semantic panel elements, and the choice of zoom factor for the virtual camera. This paper describes the mechanisms that Comic Chat uses to perform this automation, as well as novel aspects of the program's user interface. Comic Chat is a working program, allowing groups of people to communicate over the Internet. It has several advantages over other graphical chat programs, including the availability of a graphical history, and a dynamic graphical presentation.

CR Categories: I.3.2 [Computer Graphics]: Graphics Systems—distributed /network graphics; K.8.1 [Personal Computing]: Application Packages; D.2.2 [Information Interfaces and Presentation]: Tools and Techniques—user interfaces.

Keywords: Non-photorealistic rendering, comics, chat programs, virtual worlds, graphical histories, automated presentation, illustration, user interfaces, Internet, World Wide Web.

1. INTRODUCTION

Computers are rapidly changing the way that people communicate, as is evidenced by the proliferation of electronic mail and the advent of the World Wide Web. As the Internet and on-line services have grown in popularity, so has another form of computer-assisted communication: electronic chat rooms. Although electronic chat programs originally supported only text-based communications, fast modems and network connections have enabled the authors of these programs to create a richer user experience through the addition of graphics [14, 16, 22]. Participants in these new chat rooms communicate with one another not only by typing text, but also by changing the gesture or facial expression of an *avatar*, which serves as their graphical representation. In these chat rooms, graphics serves as a first-class form of communication—it is essential, not merely ornamental.

Considering the importance of graphics to modern chat rooms, it is a true failing that these programs provide only textual history transcripts. Histories are crucial in chat programs, since participants commonly interleave several tasks while chatting on-line. For example, people often participate in chats while writing documents, taking care of children, compiling programs, or monitoring other chat rooms. Any graphical communication that occurs while a participant is tending to another task is either lost or translated into textual form. Since there is no graphical history of the conversation, it is difficult to share the experience with non-participants, or to review the chat at a later time.

A second problem with current graphical chat programs is the requirement that participants must spend a significant amount of time doing things other than chatting. Most graphical chat programs require that participants navigate the room (or world), looking for an interesting conversation. Gestures must be specified either by hitting memorized function keys, or navigating menus or buttons. Care must be taken not to obstruct the view of other avatars, or to keep them in view. In 2D chat rooms avatars often overlap, and in 3D rooms they can accidently move behind or between other participants in the conversation.

A third aspect of graphical chat programs that can be improved upon is the relatively static nature of the graphical presentation. Although some chat rooms have animated backgrounds, the participants tend to see the same exact scene composition, at the same exact point of view, unless they or others move. Much can be done to make the view and scene composition more dynamic, and to make them reflect and enhance aspects of the conversation.

To address these problems, we have developed a new graphical representation of electronic chat rooms, based on the visual conventions of comics. Comics have a very rich tradition, and a distinctive, compelling, and entertaining visual vocabulary. Nearly everyone is familiar with the comic form, and many of its conventions have become second nature, even to the few that rarely read comics. Many of the same visual conventions are shared by comics throughout the world, which is important when choosing a graphical presentation for chat rooms on worldwide electronic networks.

We have built a system, called "Comic Chat," that automatically generates comics to depict on-line graphical chats. Relying on the rules of comic panel composition, this system chooses which avatars (presented as comic characters) to include in each panel, determines their placement and orientation, constructs word balloons of multiple types, and places these word balloons according to the rules for proper reading order. The system also chooses an appropriate camera zoom factor for each panel, decides when to begin a new panel, and selects default gestures and expressions that the participants can override. To make the chat rooms more lively and fun, Comic Chat can adjust the background or scene elements to reflect the topic of the conversation. All of the comics appearing in this paper were generated automatically by the Comic Chat system. In several instances, we felt it important to show examples of what can go wrong in constructing comics, and in these cases we generated comics by disabling components of the system.

Although the origins of comics are very old, this project was undertaken to coincide with the centennial of the modern comic strip, as commonly measured from the publication of Richard Outcault's "Yellow Kid," first appearing in 1895 in the New York World [10]. This past year, the comic strip centennial has been celebrated in the U. S. with a set of commemorative stamps, the opening of the National Gallery of Caricature and Cartoon Art, and even a series of textual chat room discussions with professional comic artists [11]. We consider it a fitting time to explore how on-line communications might benefit from the comic strip.

The next section describes related work. Section 3 explains the methodology that we used to select the features of comics to automate in our system. Section 4 describes how Comic Chat decides which characters to include in each panel, and how it positions and draws them. Issues related to balloon construction and layout are covered in Section 5. Section 6 addresses other important considerations in constructing panels. Section 7 discusses implementation issues. Several examples of our system's output are provided in Section 8. Finally Section 9 presents our conclusions and describes possible directions for future work.

2. RELATED WORK

Some of the best analyses of the visual language of comics have been written by comic artists. Both McCloud [13] and Eisner [5] authored excellent books on the subject. Several comics have been published electronically, some of which use techniques like color-table animation and staged presentation, to achieve effects beyond the scope of traditional comics [4]. Other commercial software programs allow people to author their own comics [15]. However, these programs are essentially drawing programs that allow users to assemble comics from clip art—they do not attempt to automate the process of comics production.

Several computer graphics researchers have built systems to produce illustrations automatically that satisfy a specified goal. Feiner's APEX system generates 3D illustrations depicting the steps of tasks, such as equipment repair [6]. Mackinlay's APT system determines the form of graph that best represents a given type of quantitative information, selecting which graphical attributes are best suited to represent that information [12]. The IBIS system, developed by Seligmann and Feiner, employs a generate-and-test mechanism with backtracking, to create illustrations that satisfy a given communicative intent [19]. Comic Chat also automates the construction of illustrations, but it is more special purpose, relying on domain-specific techniques.

Cassel et al. developed techniques to automate gestures and facial expressions in the animation of conversations between animated characters [3]. Their system relies on a database of facts, goals, and beliefs, not only to generate aspects of the characters' animations, but also to generate the text spoken by the characters. Comic Chat also generates default gestures and facial expressions for its characters, but because the system receives unconstrained textual input, it is difficult to produce sophisticated inferences about the conversation to control gesture and expression. Instead, Comic Chat uses simple rules, based on comics and on-line chat conventions, to choose default gestures and expressions, which can be overridden by the user.

Research aimed at achieving aesthetic, non-photorealistic images is receiving increasing interest in the graphics community. Haeberli developed techniques to produce artfully painted pictures interactively from image data [8]. Winkenbach and Salesin [21] and Salisbury et al. [18] devised methods for generating pen and ink illustrations from 3D models and images. Today, comics are also gaining recognition as an artform, and although a few comic artists are beginning to use computer-based tools, comics have not previously been targeted for computer graphics research.

A benefit of depicting chat sessions in comics form is that the primary visual representation also serves as a history transcript. Kurlander's editable graphical histories also represent the history of graphical user interaction as a series of panels [9]. However, that history representation adopts no other conventions from the visual language of comics.

Comic Chat was motivated by the appearance of several graphical chat programs in recent years. Habitat, built by members of Lucasfilm's game group [16], is a virtual multi-user world, rather than simply a chat system, although providing a new means for social interaction and communication was one of the project's major goals. Habitat places a very simple word balloon over the head of the last avatar to speak. When the next avatar speaks, the earlier word balloon loses its tail and scrolls upwards. As avatars move about, their balloons do not follow. Balloon construction in Habitat is very coarse, only slightly reminiscent of comics, and no real layout is performed. Yet Habitat was revolutionary in showing that on-line graphical virtual communities are not only possible but worthwhile, and later incarnations of the system are popular today, including Fujitsu Habitat in Japan, and CompuServe's WorldsAway.

More recently, several companies have deployed three-dimensional graphical chat rooms. In Worlds Inc.'s WorldsChat, participants navigate a virtual space station, and see the environment from either a first-person or third-person perspective [22]. All communication, however, is textual. On the Microsoft Network, V-Chat (for "Virtual Chat") provides several 2D and 3D environments. Participants can gesture and change their avatars' expressions, as well as type text to one another [14].

3. METHODOLOGY

To help us determine how to best represent chat sessions in the comic form, we first gathered numerous chat transcripts from an on-line service. We then read through the transcripts and annotated them with any information that we felt a computer would be able to extract easily, and that might be useful in composing the strips. Jim Woodring, a professional comic artist, took these transcripts, and illustrated a representative session.

Prior to seeing Woodring's illustrations, we were concerned that an effective comic chat representation might need to reflect deep semantics of the conversation, semantics beyond the reach of current natural language understanding technology. After reviewing the prototype illustrations, it became clear that we could produce interesting comics with only very limited semantics, and without any natural language processing.

Comic styles vary so dramatically that a single system capable of producing them all is inconceivable. Instead, we undertook to mimic the style of Woodring's prototype illustrations as precisely as possible in the Comic Chat system, and we designed the architecture so that it could be readily extended to include additional styles.

Carefully reviewing the artist-drawn prototypes, we determined the design elements that require automation. These elements fall under three broad categories: *characters*, *balloons*, and *panels*. The following three sections address each category in turn.

4. CHARACTERS

In our system, comic characters serve as avatars for the chat participants. Participants select their own character from a presupplied list. In composing the panels, several aspects of the characters need to be determined, including their gestures and expressions, which characters to draw, and their positions and orientations.

4.1 Gestures and expressions

We refer to *gesture* as a character's body pose, and *expression* as its facial pose. Most of the characters in Comic Chat have a set of heads that can fit on a set of bodies; hence their expressions can be independent of their gestures. Since one of our design goals is to minimize the amount of user interaction necessary, Comic Chat picks default gestures and expressions, which the user is free to override.

When a participant types in dialogue for his or her character, Comic Chat uses this dialogue to determine a default gesture and expression. An analysis of chat transcripts revealed a number of chat-specific conventions that we could effectively exploit, and our comic artist suggested a few additional indicators:

1. **Emoticons:** for example, :-) and :-(, indicate happy and sad.
2. **Chat acronyms:**
    a. LOL (Laughing Out Loud), ROTFL (Rolling On The Floor Laughing) indicate laughter
    b. IMHO (In My Humble Opinion) results in character pointing to self.
    c. BRB (Be Right Back) results in character waving.
    d. <g> (or <grin>) indicates smiling.
3. **Typesetting:** All caps indicates shouting.
4. **Punctuation:** !!! indicates shouting.
5. **Greetings:** Hi, hello, bye, goodbye, welcome (at beginning of sentence), result in character waving.
6. **Self-references:** I (at beginning of sentence), I'll, I will, I'm, I am, I'd, I would, etc... result in character pointing to self.
7. **Other-references:** You (at beginning of sentence), are you, will you, did you, don't you, etc... result in character pointing to other.

Initially, we had thought that simple punctuation would be a much better indicator for expressions in comics (for example, a "?" for questioning, or a single "!" for exclaiming). However, typically punctuation is an indicator of fairly subtle expressions, and such subtlety tends to get lost in comics. In contrast, characters pointing and waving, which occur relatively infrequently in real life, come off well in comics.

Note that since gesture and expression can be controlled independently, multiple indicators can be extracted and used from the same input. For example, "I can't make it :-(", results in a character pointing to itself and frowning. Since multiple indicators can be extracted from a single input, they may also conflict. For example, "Hi Sue, how are you?", would suggest that the character both wave and point outward. Since constructing composite gestures from many individual 2D bitmaps of body parts (left arm pose, right arm pose, etc...) makes for a nearly impossible art authoring process, Comic Chat instead uses a prioritization scheme to choose the most important gesture.

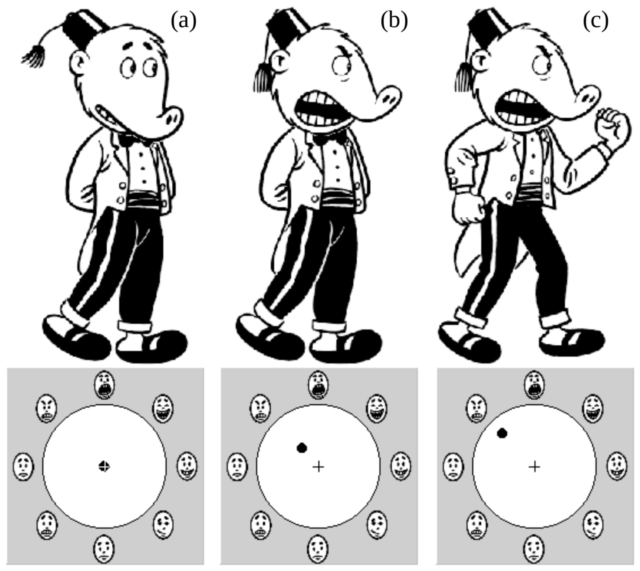


Figure 1. Emotion wheel and three bodies. a) a neutral pose; b) somewhat angry; c) very angry.

When no expression or gesture indicators are found in the utterance, the system chooses a default neutral expression and gesture. The system has multiple neutral expressions and gestures, and cycles through these to increase the visual richness of the comics produced. Often there are multiple versions of other expressions and gestures, too. Participants can see the system-selected expression and gesture for their character as they enter text. If they would prefer to override these, they can do so with a user-interface widget that we call an *emotion wheel*.

The emotion wheel is a circle, with emotions distributed along its perimeter: coy, happy, laughing, shouting[1], angry, sad, scared, and bored. At the center of the circle is neutral, and the further from the center, the greater the intensity of the emotion. The emotion wheel is similar to color choosers that distribute fully-saturated hues along the perimeter and have a shade of gray at the center. A nice aspect of the emotion wheel is that although emotion type and intensity are controlled with a single mouse movement, the results are still very intuitive.

Figure 1 shows the emotion wheel, along with the character feedback pane from Comic Chat's user interface. Initially, in Figure 1a, the control is at the center, and the character is in a neutral pose. In Figure 1b, the character becomes slightly angry, as the control is moved towards the "angry" direction on the perimeter. The character becomes much angrier in Figure 1c, as the control is moved further toward the perimeter. Emotions can affect both expression and gesture, as seen in Figure 1. Having both expression and gesture contribute to the portrayal of an emotion can provide many more shades of an emotion, given a limited supply of art.

In comics as in life, intense emotions can turn to actions, so it may make sense to add actions to the emotion wheel as well. For example, if the character becomes extremely angry, he or she might punch another. While this may be amusing at first, we fear it would become overly disruptive.

Actions that do not correspond to emotions, such as waving and pointing, are not accessible from the wheel. Instead, users can invoke them by mousing on the target character in any panel, and

1. Technically, shouting is not an emotion, but an action. However, we have found that it works best to include it on the wheel.

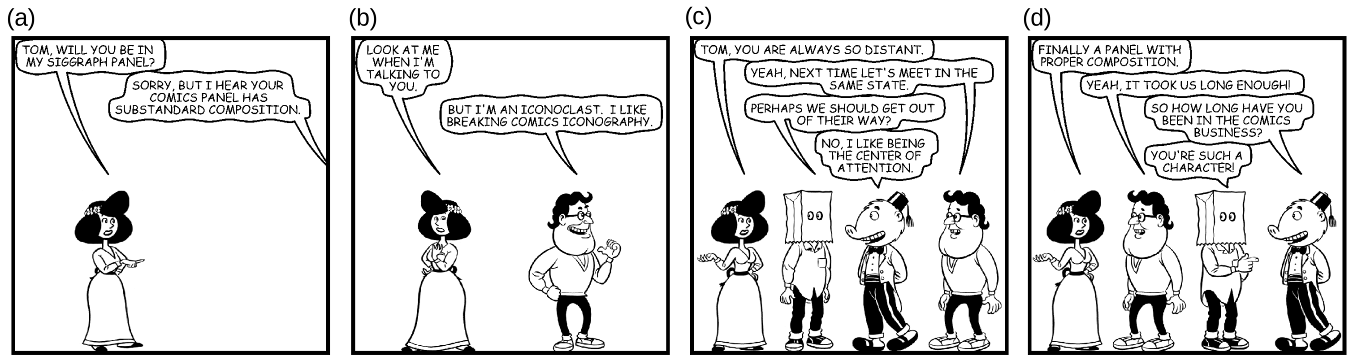

(a)
TOM, WILL YOU BE IN MY SIGGRAPH PANEL?
SORRY, BUT I HEAR YOUR COMICS PANEL HAS SUBSTANDARD COMPOSITION.
(b)
LOOK AT ME WHEN I'M TALKING TO YOU.
BUT I'M AN ICONOCLAST. I LIKE BREAKING COMICS ICONOGRAPHY.
(c)
TOM, YOU ARE ALWAYS SO DISTANT.
YEAH, NEXT TIME LET'S MEET IN THE SAME STATE.
PERHAPS WE SHOULD GET OUT OF THEIR WAY?
NO, I LIKE BEING THE CENTER OF ATTENTION.
(d)
FINALLY A PANEL WITH PROPER COMPOSITION.
YEAH, IT TOOK US LONG ENOUGH!
SO HOW LONG HAVE YOU BEEN IN THE COMICS BUSINESS?
YOU'RE SUCH A CHARACTER!

Figure 2. Three poorly composed panels, and one well-composed panel. a) missing speaker; b) conversational participants face away from each other; c) outer conversational pair is separated; d) well-composed panel.

then choosing the action from a pop-up menu. Since much of communication is non-verbal, Comic Chat allows gestures and expressions to be transmitted with or without accompanying text.

4.2 Characters for inclusion

Only about five characters can fit in one of our panels, and as in television and film, it is best not to show every character in every shot. It is, however, critical to show characters whenever they are speaking. Professional comic artists break this rule extremely rarely, and then typically only to make a stylistic statement. Figure 2a shows an example of breaking this rule.

If a character starts speaking to someone new, it is also important to include the character (or characters) being addressed. Once the addressee has been established, he or she can optionally be omitted from subsequent panels. When a participant reacts to what is said with either a gesture or expression, the system must show this as well.

When new characters enter the chat world, they are shown immediately, whether or not they say something. This makes it clear to the other participants that they have someone new monitoring their conversation, as well as someone new with whom they may wish to speak.

4.3 Position and orientation

Once it is determined which characters should be included in the panel, the next step is to position the characters and determine the direction they should face. In comics, as in life, people talking to each other tend to face one another and be situated together. Hence, the characters in Figures 2b and 2c are poorly arranged, and Figure 2d presents a better configuration. Of course, this rule suggests that it is important to know who is speaking to whom when rendering a conversation as comics.

Since the intended recipient of an utterance is not always apparent, participants in textual chat rooms often follow the convention of addressing each of their statements to particular individuals. Comic Chat looks for the nicknames of each of the chat participants in every utterance and tries to infer who is being addressed. (The nickname of a character appears whenever the cursor is placed over a character.) Alternatively, the user can also make the addressees explicit by selecting them with the mouse. Since there is a great deal of coherence in who is talking to whom in a conversation, this is not a significant hassle for the participants, and it also allows us to provide an option to hide all utterances not explicitly addressed to a specific participant. If someone chooses not to address their comment to specific individuals, it is interpreted as a general statement to the group, and the character is ideally made to face as many participants as possible.

We have constructed an evaluation function that considers these criteria, as well as some additional ones, and returns a lower value for better layouts. Keep in mind that the actual numbers in this function are arbitrary, but the function does capture the criteria we feel are important as well as the relative significance of the criteria. Let C be the set of characters to be included in the panel. Let a and b range over every member of C. We search for an ordering and orientation of the characters that minimizes the sum:

$$\sum_{a,\, b \in C \,|\, a \neq b} (Facing\,(a, b) + Neighbors\,(a, Left\,(a), Right\,(a)))$$

The *Facing* function penalizes conversational pairs in which the characters do not face one another or are not proximal. It is the sum of the numbers on the left whose predicate on the right is true:

4:	if a has not addressed his utterance, and is not facing b.
2:	if a has not addressed his utterance, and b is not facing a.
4:	if a has addressed his utterance to b, and b is not facing a.
40:	if a has addressed his utterance to b, and a is not facing b.
4n:	if a has addressed his utterance to b, and the number of characters between them is n.

The *Neighbors* function discourages the system from shuffling the positions of characters from panel to panel. For each character in the sequence, we consider the characters to the left and the right, and for each of these that is different from the character last appearing there, a single point penalty is assessed. Note that maintaining consistency of positions from panel to panel is the least important consideration, and some comic artists do not feel this to be important at all.

Now that we have an evaluation function, we try to find the positions and orientations of the characters that minimize it. Instead of trying every combination, we use a greedy algorithm. After placing the first character, we find which of the two possible positions (and two possible orientations in each position) is best for the second. After placing the second, we consider the three possible positions for the third, and so on. Although this technique is not guaranteed to find the best possible arrangement, it tends to do an adequate job and is fast.

4.4 Rendering

The process of rendering a character consists of composing together a head bitmap with a body bitmap at the proper offset, and flipping them if necessary to make the character face the proper direction.

There is however one subtle yet important detail. Comic artists that draw complicated backgrounds often leave a little white space around the character. This makes the character much more visible,

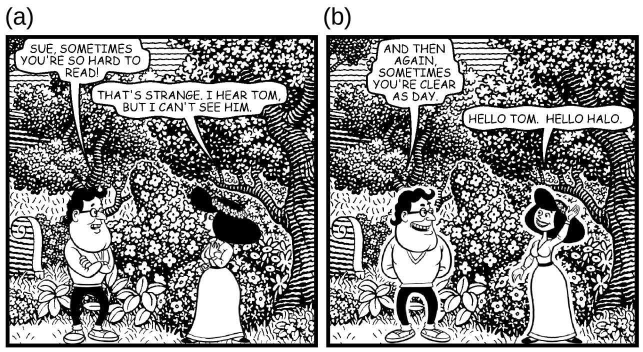

(a)
SUE, SOMETIMES YOU'RE SO HARD TO READ!
THAT'S STRANGE. I HEAR TOM, BUT I CAN'T SEE HIM.
(b)
AND THEN AGAIN, SOMETIMES YOU'RE CLEAR AS DAY.
HELLO TOM. HELLO HALO.

Figure 3. Halos. a) characters without halos; b) characters with halos, improving visibility.

and makes it "pop" out of the background. We call this area around the character, the *halo*. Figure 3a shows two characters drawn without halos, and Figure 3b shows two characters with halos.

For each head and body bitmap, we also save a halo mask. It is important to draw both of the halos before drawing either the head or body, since the head's halo must not overwrite any of the body, and the body's halo must not overwrite any of the head.

5. BALLOONS

Word balloons are one of the most distinctive and readily recognizable elements of the comic medium. Most comic artists represent dialogue in word balloons, and these balloons are clearly critical in Comic Chat, since much of the communications between participants is still textual. This section addresses the various types of word balloons, as well as issues in their layout and construction.

5.1 Types

The appearance of balloons varies dramatically from artist to artist. Some artists draw balloons as ovals, others draw rectangular balloons. Certain artists omit the traditional balloon outline, and draw a dash from the character to the words. Woodring draws balloons that flow around his text. We felt that these balloons would be ideal for Comic Chat, because every balloon is different, and the resulting variety and visual richness assist in hiding the machine-constructed origin of the panels.

Even for a given artistic style, there are several types of balloons within the comic vocabulary. Figure 4 shows three different balloon types in the style that we are emulating. The most basic type is the speech balloon, an example of which appears in Figure 4a. The outline is solid, with the upper part (the *body* of the balloon) surrounding the dialogue. Emanating from the body is the *tail*, which points towards the speaker. Figure 4b illustrates a thought balloon. It is similar to the speech balloon, except instead of having a solid tail, its tail is composed of ovals. In some styles, the body of the thought balloon appears more like a cloud.

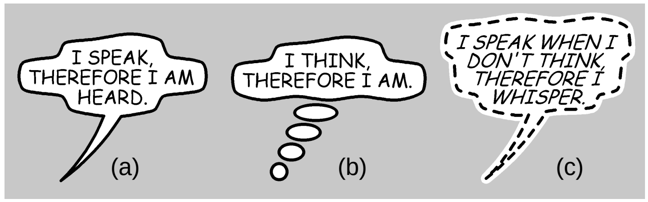

I SPEAK, THEREFORE I AM HEARD.
(a)
I THINK, THEREFORE I AM.
(b)
I SPEAK WHEN I DON'T THINK, THEREFORE I WHISPER.
(c)

Figure 4. Three types of balloons: a) speech balloon; b) thought balloon; c) whisper balloon.

Figure 4c shows a whisper balloon. It differs from speech balloons in having a dashed outline, which is made more visible by the presence of a halo. The text in whisper balloons is often italicized. In chat terminology, *whispering* means sending a message only to a select subset of the participants, and it is convenient that the comic vocabulary already has a convention for this. A fourth type of balloon is the shout balloon, which has a jagged outline (and is yet to be implemented in Comic Chat). Although other types of balloons do exist, these four are the most prevalent.

5.2 Layout

Before constructing balloons, we run a layout algorithm that determines each balloon's dimensions and placement. In the style that we are emulating, all balloons appear above the tallest character's head. However, additional constraints must be considered as well.

In comics, the relative placement of balloons determines the reading order. Because it is important in Comic Chat to convey the order of the utterances, we strictly follow the appropriate comic conventions. The rule is that balloons are read in a top-down order, and when multiple balloons are at the same height, they are read from left to right. Note that the order in which balloons are read is independent of where characters are placed in the panel.

A third point that helps constrain the placement of the balloons in Comic Chat is the style-specific requirement that some part of each balloon float at least partially over the center of the speaker's face. We have designed a layout algorithm that takes all of these rules into account, while adding a little randomness to each layout to avoid the appearance of machine-generated regularity.

We use a greedy algorithm for placing the bodies of balloons; once the algorithm chooses a body location, it never alters it. However, the tail placement is not greedy; instead, the algorithm defers placing the tails until all the bodies have been placed. Note that using a greedy algorithm is suitable for body placement because it is fast, does a good job, and because there is no real need to pack the maximum amount of text into a panel, as cartoonists do not typically do this either. On the other hand, we originally tried using a greedy algorithm for tail placement, with poor results.

Our layout algorithm makes use of intervals, called *routing channels*, for ensuring that there will be enough room to place the tails once the bodies are positioned. Routing channels are disjoint—they are a partitioning of the space available for routing balloon tails. Each routing channel is associated with a balloon and indicates where that balloon's tail could be placed. For example, in Figure 5a, the darkly shaded area is the original routing channel for the topmost balloon, spoken by the center character. This region spans the width of the balloon; initially, the tail could go anywhere in this region and be guaranteed not to intersect another balloon or tail. When a second character speaks (in this case, the leftmost character), the first routing channel is decreased in size in order to accommodate the second balloon, as shown in Figure 5b. The third balloon, in its original position, completely covers what remains of the first routing channel, leaving no place for the first balloon's tail. Our algorithm recognizes this situation and shifts the third balloon right in order to leave enough space for the first balloon's tail. The final routing channel for all three balloons is shown in Figure 5c.

Now that we have run through this example, let's define the routing algorithm more formally. To describe our algorithm, we will first introduce some notation. Let $B = (B_1,...,B_n)$ be an array of balloons in the order in which they appear, with $T = (T_1,...,T_n)$ representing the text in the balloons. Let $R = (R_1,...,R_n)$ be a corresponding array of routing channels, with each routing channel R_i specified by left and right endpoints $R_i.l$ and $R_i.r$. Each balloon B_i is spoken by a

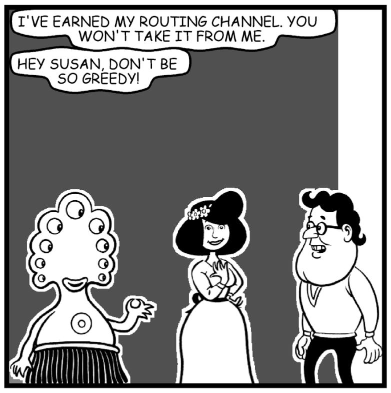
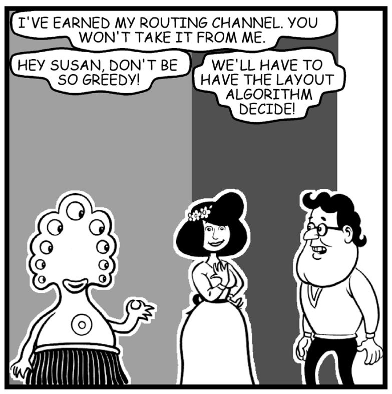
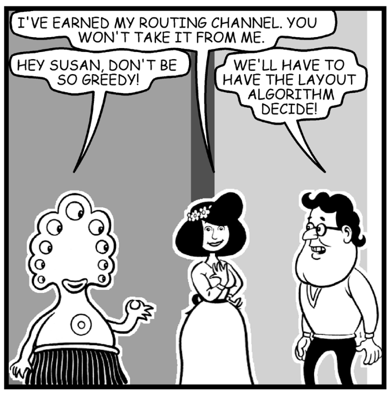
(a) 
I'VE EARNED MY ROUTING CHANNEL. YOU WON'T TAKE IT FROM ME.
HEY SUSAN, DON'T BE SO GREEDY!
(b) 
I'VE EARNED MY ROUTING CHANNEL. YOU WON'T TAKE IT FROM ME.
HEY SUSAN, DON'T BE SO GREEDY!
WE'LL HAVE TO HAVE THE LAYOUT ALGORITHM DECIDE!
(c) 
I'VE EARNED MY ROUTING CHANNEL. YOU WON'T TAKE IT FROM ME.
HEY SUSAN, DON'T BE SO GREEDY!
WE'LL HAVE TO HAVE THE LAYOUT ALGORITHM DECIDE!

Figure 5. Routing Channels. (a) Lower left balloon requests channel allocation from top balloon; (b) Lower right balloon requests channel allocation from top balloon; (c) Lower right balloon shifts right, routing regions successfully allocated.

character, and the x coordinate of the center of that character's face is represented by x_i.

The following procedure computes the horizontal placement for an array of n balloons, and sets up the routing channels $R_1,...,R_n$. It returns the number of balloons that is placed successfully.

```
function PlaceBalloons(B, R, x, T)
    for $j$ = 1 to $n$ do
        $w_j := FindWidth(B_j)$;
        $R_j := [x_j - w_j, x_j + w_j]$
        for $i$ = 1 to $j$-1 do:
            $R_j := MaxAllowable\ (R_i, x_i, R_j, x_j)$;
        end for
        if width $(R_j) >= w_j$ then
            $R_j := Position\ (B_j, R_j)$
        else
            if not $SqueezeBalloon\ (R_j, T_j)$ then
                return $j$-1;
            end if
        end if
        for $i$=1 to $j$-1 do
            $R_i := ReduceChannel\ (R_i, x_i, R_j, x_j)$
        end for
    end for
    return $n$
end function
```

The *PlaceBalloons* function loops through each balloon, placing them one at a time. Let us assume that balloons B_1 through B_{j-1} have been placed, and let's look at how to place balloon B_j. The first step is to choose a target width for the balloon (described in detail later), and set the largest possible routing channel that would allow the balloon's horizontal extent to pass above the center of the character's face.

Next, we trim the routing channel R_j just enough to ensure that all of the previous routing channels $R_1,...,R_{j-1}$ remain wide enough to support a tail. If the resulting routing channel R_j is wider than the target width, then we have to choose a horizontal position for the balloon in the routing channel (we do this randomly); we then set the routing channel to be the horizontal extent of the balloon. Otherwise, we attempt to squeeze the balloon into a narrower channel than the original target width. If the amount of text in the balloon is too large to fit, then we give up and return the number of balloons previously placed. Finally, we reduce all of the previous routing channels to accommodate the new one.

This describes the heart of the algorithm. All that is left are the details. To choose the target width, we initially estimate the area that the body will cover, by computing the area of a single typeset line, and scaling it up by a factor of a third to account for linebreaks and leading. If the line is short, it should not be broken into multiple lines, and we assume a height of one text line. Otherwise, we determine a conservative estimate of the allowable height of this balloon by calculating the distance from the bottom of the lowest previously placed balloon to the bottom of the rectangle in which balloons can be placed. The minimum allowable width of the balloon text is the maximum of the widest word, and the balloon area divided by the allowable height. We randomly select a width between this minimum and the width of the panel.

The function below is used to trim the routing channel R_j just enough to ensure that routing channel R_i remains wide enough to support a tail. It ensures that the width of routing region R_i is at least t, and that R_i continues to contain x_i:

```
function MaxAllowable $(R_i, x_i, R_j, x_j)$:
    $R := R_j$
    if $x_i < x_j$ then
        $R.l := \max\{R_i.l + t, x_i\}$
    else
        $R.r := \min\{R_i.r - t, x_i\}$
    end if
    return $R$
end function
```

Finally, the *ReduceChannel* function reduces the interval R_i so that it no longer overlaps R_j:

```
function ReduceChannel $(R_i, x_i, R_j, x_j)$
    $R := R_i$
    if $x_i < x_j$ then
        $R_i.r := \min\{R_i.r, R_j.l\}$
    else
        $R_i.l := \max\{R_i.l, R_j.r\}$
    end if
    return $R$
end function
```

As mentioned earlier, the above algorithm finds the horizontal placement of a set of balloons. Next we calculate the vertical placement. The rules for proper reading order require that the new balloon be no higher than the bottom of any balloons already placed to its right, and no higher than the top of any balloons already placed to its left. We place each balloon as high as possible, such that proper reading order is maintained.

If the function fails, at least one balloon could not be successfully placed in the current panel. We check to see if the balloon can fit in a panel by itself. If it cannot, we break up the text of the balloon into smaller balloons that do fit the panel size constraint, and add ellipses to each of the new balloons to indicate that a split occurred.

5.3 Balloon body construction

Since the bodies of our balloons flow around the text that they contain, the first step of constructing balloons consists of computing the layout of this text. The artist we are emulating typically centers text in word balloons, so we first compute word breaks and then center the text on each line.

Because our goal is to create balloons that wrap smoothly around text, it makes sense to construct the bodies of these balloons with splines. After experimenting with cardinal splines and B-splines, we chose the latter, since they produce the best effect. Our artist draws moderately sharp bends in his balloon outlines, which we were able to mimic using a high B-spline tension of 5.0. When comic artists draw balloons, they always leave a margin between the text and the balloon outline, so we expand the boundaries of the text lines outward before computing the spline.

Our original attempt at balloons appeared amoeba-like, following the text too closely, responding to its every turn. We prefer balloons that flow around the text, but do not draw too much attention to themselves. In studying a set of sample balloons drawn by our artist, we determined that he avoids this problem in two ways. First, he never dips the balloon inwards towards the text on one line, just to move outwards again on the next. Second, he tends not to shift the balloon outline inwards or outwards by very small increments—instead he responds only to larger changes in the text outline.

We implemented these two additional rules, and the resulting balloons were much improved, yet still inferior to balloons that we wished to emulate. The large remaining difference between our balloons and Woodring's was the lack of small-scale perturbation in the outline. Whenever there is a long segment of the outline that does not bend in or outward with the text, Woodring adds small, low frequency waves to give the balloon additional richness. We have been able to mimic this effect by placing additional control points along these large segments, which alternately move towards and away from the text. The resulting balloons produced by Comic Chat faithfully capture Woodring's style, and examples of these balloons appear in figures throughout the paper.

5.4 Balloon tail construction

Now that we have placed and constructed the bodies of the balloons and computed routing channels for the tails, it is time to construct the tails themselves. The tails of speech and whisper balloons are composed of arcs or straight segments, and the tails of thought balloons consist of ovals forming either an arc or a line.

The best looking balloons have tails emanating from under the bottom line of text. Depending upon the placement of the characters and balloons in a panel, it is sometimes necessary to choose a non-optimal tail connection point, as comic artists do from time to time. To choose the connection point of a tail, we first see if a large enough part of the last line of text spans the routing channel. If so, we choose a location further than a specified distance from the edge of the routing channel, to insure that the tail will not be exactly adjacent to other balloon bodies or tails. Failing this, we attach the tail to a piece of the balloon within the routing channel, at a small horizontal distance from the center of the speaking character's head. This allows us to give the tail a definite arc, but not an arc that will span diagonally over a large part of the illustration.

We construct balloon tails so that they come to a point at roughly the same height, (which of course must be below the lowest balloon), and always at least in the lowest third of the region reserved for balloons. Since the best balloon tails head directly and clearly toward the speaker, it is preferable not to have tail arcs cross over the head of the character responsible for the balloon. In our comic style, tails emanating from a part of the balloon to the left of the speaker curve counterclockwise and end above the center of the speaker's face. Tails beginning to the right of the speaker curve clockwise and end at the same point.

5.5 Rendering

Rendering a balloon is simply a matter of filling its interior (to hide a background that was drawn first), scan-converting its outline, and drawing the text. There are two important details worth mentioning. Whisper balloons have halos, and these are easily rendered by initially scan-converting the outline of the balloon with a thick, solid, white pen, before rendering the true outline with a smaller, black, dashed pen. Also, most comic artists draw balloon text in all-caps, with a very distinctive (though varying) comic lettering. We also display the text in all-caps, no matter how it was typed. A wide variety of comic fonts is available commercially and as share-ware.

6. PANELS

Several additional issues should be taken into account when composing and rendering panels. These include choosing when to begin a new panel, adjusting the zoom factor of the virtual camera, and adding semantic elements to the panels.

6.1 Panel breaks

As participants specify new dialogue and body poses, Comic Chat proceeds to redraw the last panel, incorporating this additional information. At some point, however, it is necessary to start a new panel, and Comic Chat has several rules for choosing when to do so.

First, Comic Chat always begins a new panel whenever the balloon layout algorithm cannot successfully fit an additional balloon in the panel.

A second rule is that Comic Chat always starts a new panel whenever the additional input would result in too many characters in the current panel. We limit the number of characters per panel to five, since facial expressions are otherwise difficult to read. In the current implementation, Comic Chat will not draw more than one balloon for a character in a single panel, so multiple utterances from a given character is also reason to start a new panel.

Third, we also start a new panel whenever rendering the new data in the current panel would result in a loss of information. For example, if a participant specifies two different facial expressions for his or her character, then these two expressions cannot be represented in the same panel.

Finally, since we believe it desirable to have the system occasionally draw a single character in a panel, we also will break to a new panel with a given probability (15%) whenever the first utterance of the panel is longer than a few words.

6.2 Camera zoom

If movies and television were always shot with the same camera parameters, they would quickly become visually tedious, and though comics technically do not have cameras, it is still important to add variety by changing the scale at which characters and the back-

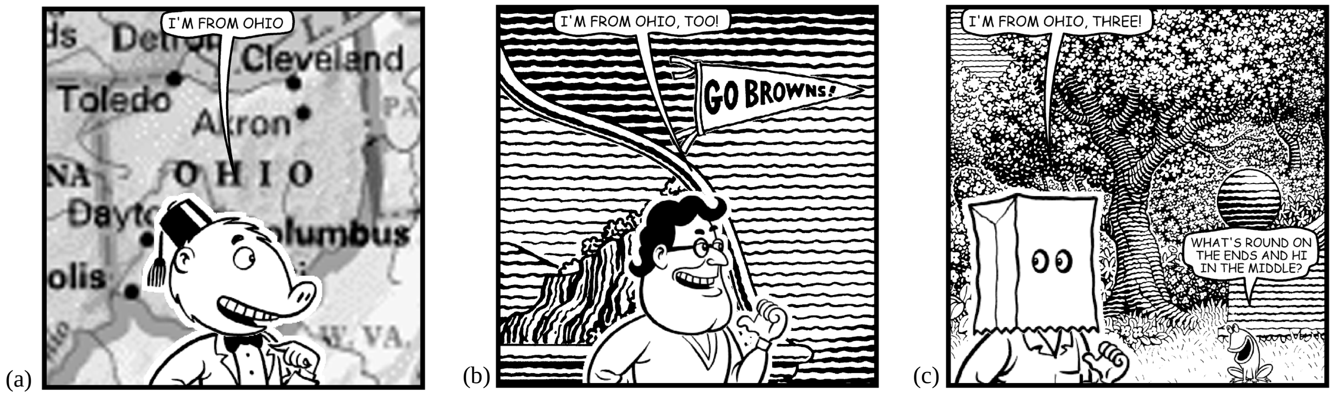


Figure 6. Semantic elements. a) map; b) scene object; c) Greek Chorus.

ground appear. Word balloons are unaffected by the virtual zoom factor.

As in the movies, it is often important to have an establishing shot. Whenever a new person enters a chat room, that person sees a panel showing the immediate surroundings. From time to time (about every 15 panels), we remind participants of their surroundings by generating another establishing shot. Other times, we pull in as close as possible, with the following caveats. First, we will not zoom in so far as to cut off a character at the neck; it nearly always looks better to include part of the character's shoulders. Second, we will not let an important character (one that we decided earlier must be included in the panel) be cut by the sides of the panel. Third, it is also best not to cut characters off at the ankles. In such cases it is better to pull back a little, and show the characters in their entirety. Some comic artists also prefer not to cut off characters at their knees either, but this is far from a universal rule, and we currently allow it.

Subject to the above restrictions, we pull in to the tightest shot possible. Since the number of characters selected to be included in the panels varies, as discussed in Section 4.2, the scale chosen for the panels also varies, contributing to the resulting visual richness.

6.3 Semantic elements

The pictures in human-drawn comics often reflect the words and stories that the comics tell. Since it is impossible given the current state of natural language technology to extract the semantics of arbitrary chat text, it is also impossible to automatically represent the meaning of this dialogue in pictures. However, things can still be done in composing the comics panels to reflect very simple semantic elements. In fact, we have already discussed how simple gestures and expressions can be selected automatically. Comic Chat reflects semantics in additional ways.

There is a collection of topics that seem to come up repeatedly in chat sessions. For example, where people are from, what they do for a living, sports teams, pets, and kids are all very popular topics. For each of these common topics, we can define keywords to look for, which when found alter the composition of the scene. Figure 6 shows three different ways that we change the panel composition whenever "Ohio" is mentioned. In Figure 6a, the background is changed to a map of Ohio for the duration of a single panel. Comic artists often use such non-realistic, representational backgrounds [13]. A disadvantage to this approach is that it dramatically increases the amount of art necessary to run the application, but it is still reasonable if the user has a CD-ROM drive. A second approach is shown in Figure 6b, where instead of changing the entire background, a new element is added to an existing background. Here we draw a banner for the soon-to-be-former-Cleveland Browns, which appropriately states, "Go Browns!"

In Figure 6c, we introduce a little meta-character at the bottom of the panel that makes smart-allecky remarks. Our comic artist refers to him as the "Greek Chorus". Here, in response to Ohio being mentioned, he asks "What's round on the ends and hi in the middle?" This character's size and placement and its balloon position distinguish him from human participants.

The beauty of the keyword approach is that it tends to work well, independent of the meaning of the entire sentence. For example, it is fine to show a map of Ohio whether a participant said "I was born in Ohio," "I've never been to Ohio," or "My plane got stuck in Ohio." A problem with this approach is that it requires a large effort to generate enough pictures and clever remarks to have reasonable coverage.

7. IMPLEMENTATION

Comic Chat is implemented in C++ and runs on both the Microsoft Windows 95 and Windows NT operating systems. Currently Comic Chat uses the Internet Relay Chat (IRC) protocol [17], and it interoperates with the many IRC servers already on the Internet. Non-textual information, such as gesture and expression choice, is encoded as a small string at the beginning of each message. Users of existing textual chat programs can communicate with Comic Chat participants, except they receive a text-only view of the proceedings, and are represented by a randomly selected character to Comic Chat participants. An early version of Comic Chat used Microsoft Network Protocols, which allowed text to be transmitted in specially-marked text packets, and all other information to be transmitted in data packets. This enabled Comic Chat to make the non-textual information totally invisible to text clients. Unfortunately, IRC lacks this capability.

Information communicated between clients includes indices of the specific bitmaps used, as well as the symbolic gestures and expressions. When a receiving client has the same character art used by the sender, it can simply consult the bitmap indices to render the exact pose chosen by the sender. However, if the receiver does not have the same character art, the symbolic gesture and expression information is applied to a different character that does reside on the system to yield a pose with the same intent.

Comic Chat has an extensible rule set for inferring a pose from typed text. End users can define their own rules. These rules are only applied to the text that the sender types, and the resulting pose information is communicated to other clients. However, when a message is received from a text-only client, the rules are applied to that message as well to assign a reasonable gesture and pose to the text client's character.

Each Comic Chat client makes its own panel composition decisions, which are not communicated from client to client. Hence, the participants in a single conversation can see different panels representing the same communications. Although on the surface this may seem to be a poor choice, it actually makes panel display more flexible, because each participant sees a custom view of the conversation. Currently, there are several factors that affect this custom view. First, participants can customize their panel size, and since more conversation fits in larger panels, this affects panel composition. Also, because Comic Chat displays an establishing shot for new participants, the panel appearance can differ according to when the participant joined the chat. In addition, individuals can whisper to one another, and these utterances appear only to those involved.

Comic Chat works as a standalone application or in conjunction with Web browsers. People can place pointers to chat rooms on their Web pages, and clicking on such a pointer will automatically launch Comic Chat to the chat room. Because it was written as an ActiveX Document [20], Comic Chat appears as a dynamic Web page within the window frame of browsers that support this interface, such as Microsoft Internet Explorer 3.0.

Performance of Comic Chat meets our original goal of requiring less than a second to compose and draw panels on a Pentium class machine. The background and character art are kept resident on each client machine, so that they need not be transmitted over the network. Artwork for a single character can vary in size according to the size and color depth of the bitmaps, as well as the number of poses provided. We have found 50 KBytes (compressed) to be sufficient for representing a character with ten to fifteen head and body drawings. New characters and backgrounds can be loaded dynamically.

Although we focused much of our efforts on recreating a particular comics style, the system architecture was designed to allow new styles to be plugged in. Comic Chat has an object-oriented structure, and includes classes for comics pages, panels, balloons, characters, character poses, and backgrounds. Elements of our artist's style are encapsulated in sub-classes of these. However, it is important to note that large amounts of effort went into defining these sub-classes, so reproducing additional artists' styles will be easier in the future, yet far from trivial.

8. EXAMPLES

This section shows three different types of chat interactions depicted by Comic Chat. The first, Figure 7, is an ordinary chat exchange between two of the authors. It contains six panels of comics, including a title panel. The title panel lists the most active participants (both of them in this case), and this information is also available interactively. All aspects of these panels, including character placement, gesture and expression choice, balloon construction and layout, panel breaks, and panel zoom, were chosen automatically by the system.

The second example, Figure 8, shows one person's interactions with a *bot*. Bots are simple computer programs, often masquerading as humans on chat channels. In some cases they perform useful functions, in others they exist to irritate other chat participants. One conversational bot is loosely modeled after the Eliza program, originally written by Joseph Weizenbaum at MIT to mimic a human psychoanalyst [2]. By joining an Eliza room, chat users can receive free (and valueless) psychoanalytical help. Figure 8 was generated by a volunteer from a Cub Scout troop, getting a demonstration of Comic Chat. The Cub Scout appears on the left, Eliza appears on the right. We halted the demonstration after the eighth panel.

People often use chat programs for role playing and interactive fiction. Figure 9 is an excerpt from such a gathering on the Internet. The participants were all using textual chat clients, and their dialogue was rendered using Comic Chat. Note that the third panel of Figure 9 also includes a *narration box*, an additional comics element supported by our program. Narration boxes help to describe what is taking place in a panel. Balloons can be placed to the right and below the narration box, according to the algorithm presented earlier. IRC clients typically support a special kind of message to indicate what a participant is doing. Comic Chat looks for such messages, and places them in narration boxes. Other kinds of information could be placed in narration boxes as well, such as who is entering and leaving. The character wearing a tie in this figure is in charge of the role playing game, and is providing narration through dialogue that is rendered in balloons. Much of this could be placed in narration boxes to produce a different effect.

9. CONCLUSIONS AND FUTURE WORK

The panels produced by the methods described in this paper are acceptably composed according to comic conventions. Jim Woodring is very pleased with the results in general, and occasionally points out panels that (he claims) he would be proud to have drawn. However, we feel that reproducing the creativity and variability of a human comic artist, particularly one of Woodring's abilities, is in all practicality an unachievable goal. Nevertheless, people seem to take delight in the output of Comic Chat, and we look forward to performing user studies to compare Comic Chat to other graphical chat systems.

There are still numerous additional features that we would like to add to the system. Professional comic book artists (in contrast to most comic strip artists) tend to show a great deal of imagination in how they lay out panels on the page. Currently, Comic Chat places square panels regularly on an infinitely long page. To create more variability, we would like to alter the size and shapes of our panels, and lay them out more creatively. This could either be done algorithmically, or by selection from a pool of template pages.

Although we have distinct locations where characters can enter our comics world (including a room in a house, a pastoral scene, a balcony, a fantasy world, and a pond scene), we should have a mechanism to allow our characters to graphically transition from one scene to another. This could happen when a character decides to join a conversation elsewhere; however, the system itself, to add variety and unpredictability, might choose to move a conversation to a new location. We would also like to make the system capable of showing participants performing various activities, like having tea or swimming in a pond. This need not be under user control, and in fact putting it under user control might be a distraction to the primary chatting activity. Having the system control such activities would add variety and unpredictability, and give the participants even more to chat about.

One of the innovations of Comic Chat is that it provides a graphical transcript of an on-line conversation in which a person has participated. It could also provide a transcript of a conversation that a person *might* want to join. In current chat systems, after entering a chat room, prospective participants wait to see if the conversation interests them. This not only creates a distraction for the other participants, but it is potentially a waste of time if the conversation turns out not to be of interest. Instead, we could provide a graphical transcript of conversations for prospective participants to review immediately. We can take this idea further, and intersperse *meanwhile panels* during gaps in a Comic Chat conversation. These panels

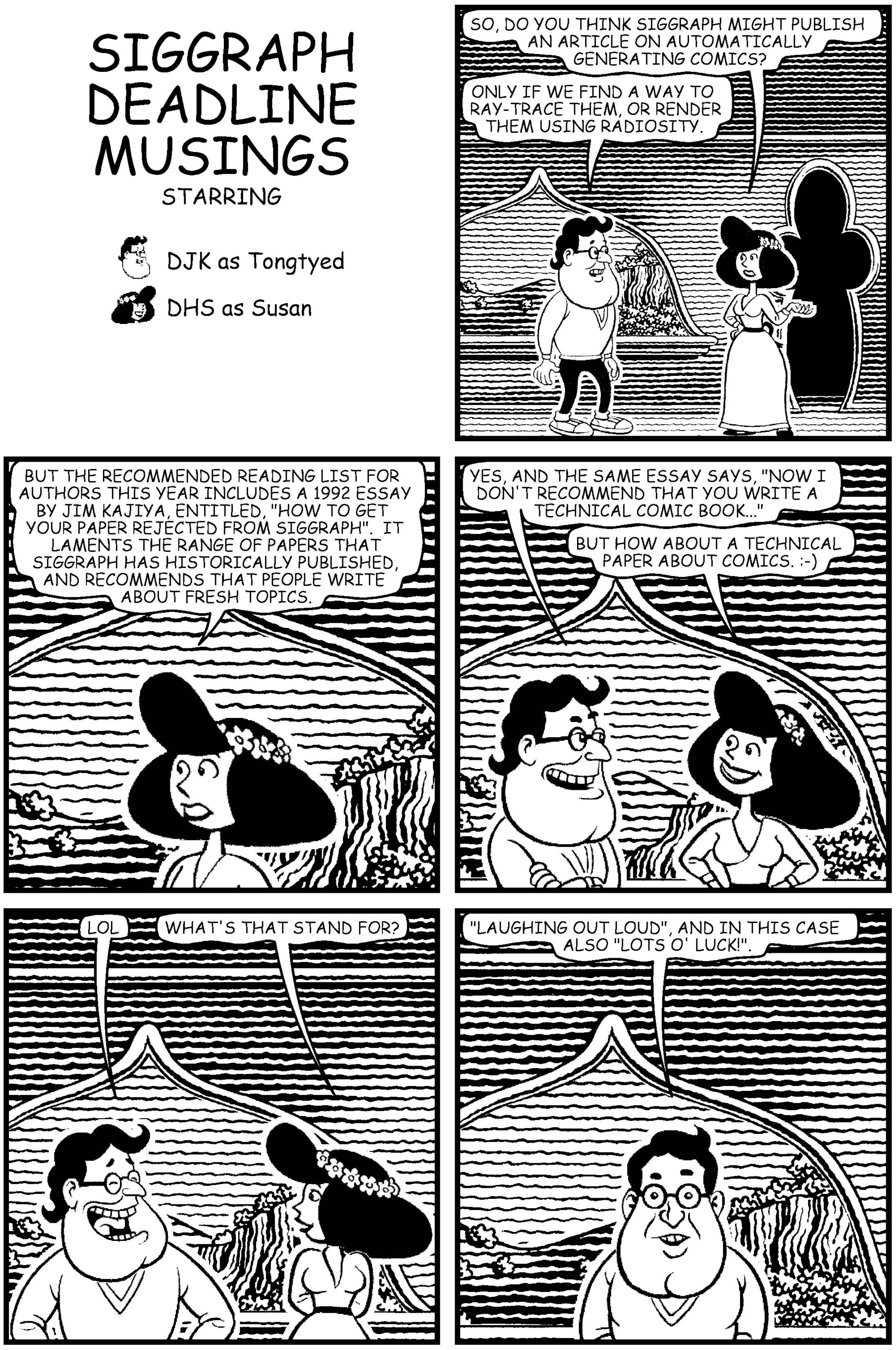

SIGGRAPH DEADLINE MUSINGS
STARRING
DJK as Tongtyed
DHS as Susan
SO, DO YOU THINK SIGGRAPH MIGHT PUBLISH AN ARTICLE ON AUTOMATICALLY GENERATING COMICS?
ONLY IF WE FIND A WAY TO RAY-TRACE THEM, OR RENDER THEM USING RADIOSITY.
BUT THE RECOMMENDED READING LIST FOR AUTHORS THIS YEAR INCLUDES A 1992 ESSAY BY JIM KAJIYA, ENTITLED, "HOW TO GET YOUR PAPER REJECTED FROM SIGGRAPH". IT LAMENTS THE RANGE OF PAPERS THAT SIGGRAPH HAS HISTORICALLY PUBLISHED, AND RECOMMENDS THAT PEOPLE WRITE ABOUT FRESH TOPICS.
YES, AND THE SAME ESSAY SAYS, "NOW I DON'T RECOMMEND THAT YOU WRITE A TECHNICAL COMIC BOOK..."
BUT HOW ABOUT A TECHNICAL PAPER ABOUT COMICS. :-)
LOL
WHAT'S THAT STAND FOR?
"LAUGHING OUT LOUD", AND IN THIS CASE ALSO "LOTS O' LUCK!".

Figure 7. A page from a session with Comic Chat.

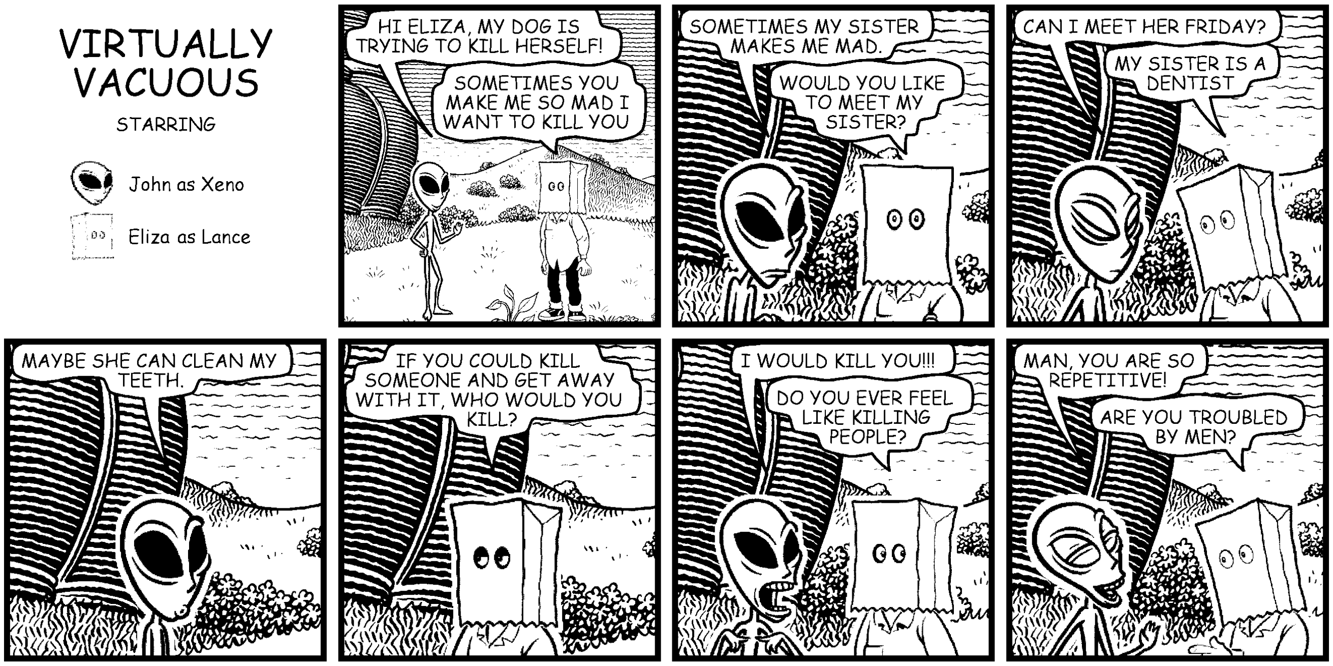

VIRTUALLY VACUOUS
STARRING
John as Xeno
Eliza as Lance
HI ELIZA, MY DOG IS TRYING TO KILL HERSELF!
SOMETIMES YOU MAKE ME SO MAD I WANT TO KILL YOU
SOMETIMES MY SISTER MAKES ME MAD.
WOULD YOU LIKE TO MEET MY SISTER?
CAN I MEET HER FRIDAY?
MY SISTER IS A DENTIST
MAYBE SHE CAN CLEAN MY TEETH.
IF YOU COULD KILL SOMEONE AND GET AWAY WITH IT, WHO WOULD YOU KILL?
I WOULD KILL YOU!!!
DO YOU EVER FEEL LIKE KILLING PEOPLE?
MAN, YOU ARE SO REPETITIVE!
ARE YOU TROUBLED BY MEN?

Figure 8. The first panels from a chat session with an Eliza bot.

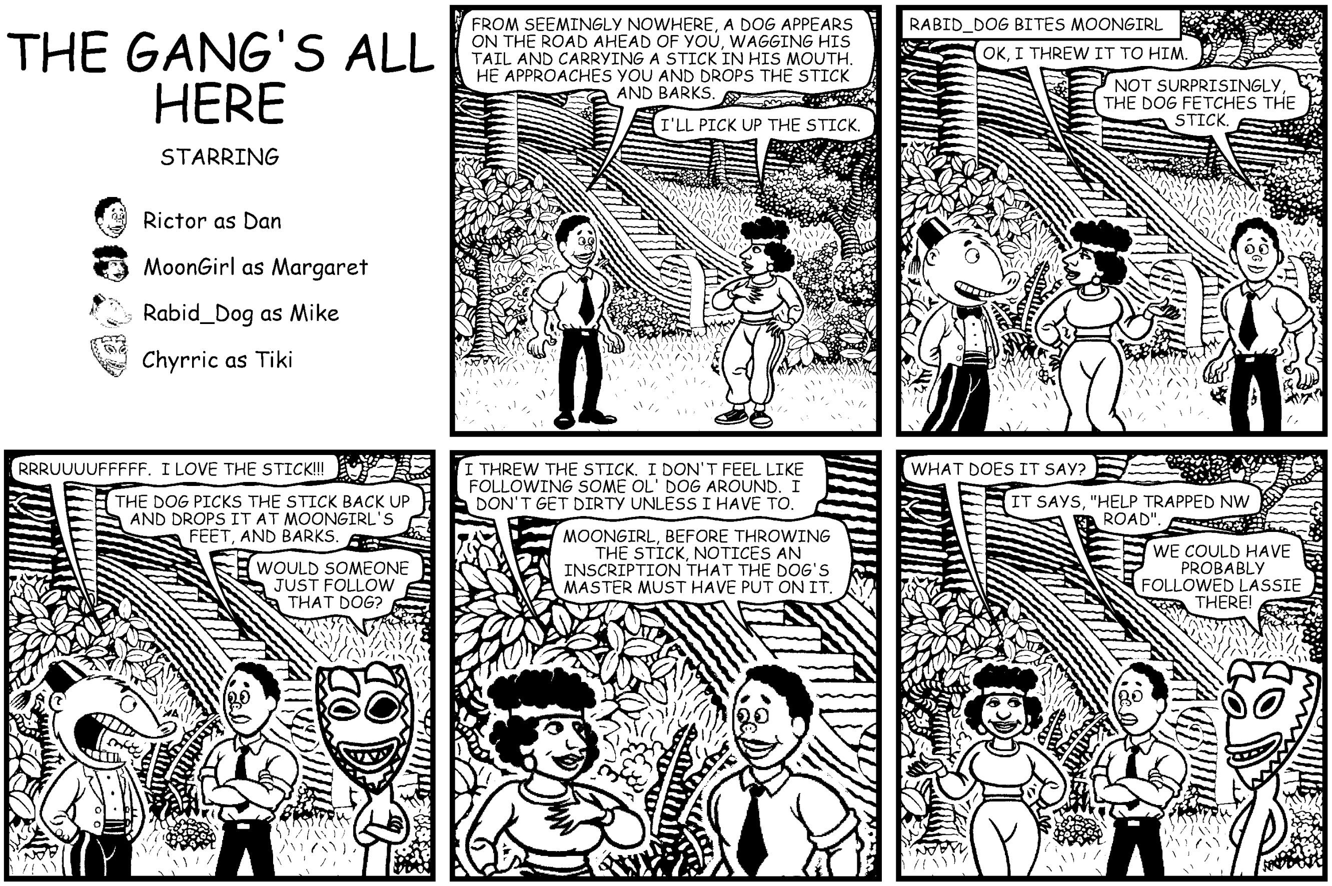

THE GANG'S ALL HERE
STARRING
Rictor as Dan
MoonGirl as Margaret
Rabid_Dog as Mike
Chyrric as Tiki
FROM SEEMINGLY NOWHERE, A DOG APPEARS ON THE ROAD AHEAD OF YOU, WAGGING HIS TAIL AND CARRYING A STICK IN HIS MOUTH. HE APPROACHES YOU AND DROPS THE STICK AND BARKS.
I'LL PICK UP THE STICK.
RABID_DOG BITES MOONGIRL
OK, I THREW IT TO HIM.
NOT SURPRISINGLY, THE DOG FETCHES THE STICK.
RRRUUUUFFFFF. I LOVE THE STICK!!!
THE DOG PICKS THE STICK BACK UP AND DROPS IT AT MOONGIRL'S FEET, AND BARKS.
WOULD SOMEONE JUST FOLLOW THAT DOG?
I THREW THE STICK. I DON'T FEEL LIKE FOLLOWING SOME OL' DOG AROUND. I DON'T GET DIRTY UNLESS I HAVE TO.
MOONGIRL, BEFORE THROWING THE STICK, NOTICES AN INSCRIPTION THAT THE DOG'S MASTER MUST HAVE PUT ON IT.
WHAT DOES IT SAY?
IT SAYS, "HELP TRAPPED NW ROAD".
WE COULD HAVE PROBABLY FOLLOWED LASSIE THERE!

Figure 9. The beginning of a fantasy role playing quest. Participants are using text IRC clients, monitored by Comic Chat.

would show snippets of conversations occurring elsewhere in the Comic Chat world (and be labeled "MEANWHILE...", following another comics convention). Of course, there is a potential privacy issue here that needs consideration.

Using more powerful technology to identify the semantics of the conversation would certainly allow us to generate better comics. Natural language processing systems would allow us to improve our choice of default expressions and gestures for the characters, and would allow Comic Chat to provide additional types of (and better targeted) semantic feedback beyond that described in Section 6.3. Many comic artists emphasize certain words in their balloons by setting them in bold type, and natural language technology could help select which words to make bold.

Comics are used for a wide variety of purposes other than entertainment, and computer-generated comics can be applied to far more than chat rooms. For example, human-drawn comics have instructed people about history [7] and the repair of military equipment [1]. Computer-generated comics could be used for education as well, but they could also serve as a tool in the production of interactive, collaborative fiction, a visual presentation for MUDs and other virtual worlds, and a graphical history of interactions with social or agent-based interfaces. The examples presented in Figures 8 and 9 hint at these possibilities. Automatically-generated comics could be used in computer-based help and instruction. They could also provide a graphical transcript or serve as the primary visual representation for adventure games. The leader of a large computer-supported cooperative work project tells us that he believes computer-generated comics could help people separated in space and time to collaborate on joint projects.

Comics are a rich form of communication, with a visual vocabulary that most of us have already internalized. They can also be wonderfully entertaining. We have found comics to be ideally suited as a visual representation for on-line chat rooms, but the potential application of comics to other computer-based applications is equally promising. The possibilities are vast, and we look forward to investigating other ways of combining comics and computer graphics.

A free copy of Comic Chat can be downloaded from our Web site, http://www.research.microsoft.com/comicchat.htm.

ACKNOWLEDGMENTS

Foremost, we would like to thank Jim Woodring for sharing his knowledge of comics, and creating such wonderful art for this project. Rick Rashid has been a great advocate (and occasional hacker). Thanks to George Robertson, Maarten Van Dantzich, Ben Slivka, David Thiel, Dan Ling, Jim Kajiya, and Steve Drucker for helpful comments. Comic Chat is publicly available on the Internet, largely due to the efforts of Linda Stone, Ross M. Brown, Diana Murray, Eric George, Mark Mecham, Bob Tabscott, and Albert Tan. Interaction with members of Microsoft Research and Microsoft's Virtual Worlds Group contributed to both the design and deployment of Comic Chat.

REFERENCES

1. Anderson, M. Joe's Dope: If Ya Gotta Do It... Do It Right! *PS: The Preventive Maintenance Monthly*. Issue 279, February 1976. 29-36.
2. Barr, A. and Feigenbaum, E. A. *The Handbook of Artificial Intelligence*. volume 1. William Kaufmann, Inc., Los Altos, CA. 1981. 285-287.
3. Cassell, J., Pelachaud, C., Badler, N., Steedman, M., Achorn, B., Beckett, T., Douville, B., Prevost, S., and Stone, M. Automated Conversation: Rule-based Generation of Spoken Expression, Gesture, and Spoken Intonation for Multiple Conversational Agents. Proceedings of SIGGRAPH '94 (Orlando, FL, July 24-29, 1994). In Computer Graphics Proceedings, Annual Conf. Series, ACM, New York. 413-420.
4. CD-ROMIX! Inc., *FREEX #1*. P. O. Box 2961, Torrance, CA 90509. 1993.
5. Eisner, W. *Comics & Sequential Art*. Poorhouse Press. Tamarac, FL. 1985.
6. Feiner, S. APEX: An Experiment in the Automated Creation of Pictorial Explanations. *IEEE Computer Graphics and Applications, 5*, 11. November 1985. 29-37.
7. Gornick, L. *The Cartoon History of the Universe*. Doubleday. New York. 1990.
8. Haeberli, P. Paint by Numbers: Abstract Image Representations. Proceedings of SIGGRAPH '90 (Dallas, TX, Aug. 6-10). In *Computer Graphics*, 24, 4 (Aug. 1990), ACM, New York. 1990. 207-214.
9. Kurlander, D. and Feiner, S. A History of Editable Graphical Histories. In *Watch What I Do: Programming by Demonstration*. Allen Cypher, ed. MIT Press, Cambridge, MA. 1993. 405-413.
10. Lent, J. Oh, What a Time It Was: The Early Days of the Funnies. Witty World: International Cartoon Magazine. No. 19. Summer/Autumn 1995. 16-22.
11. Lent, J. "Yellow Kid" Celebrates Century with Hectic Schedule in 1995. Witty World: International Cartoon Magazine. No. 19. Summer/Autumn 1995. 18-19.
12. Mackinlay, J. Automating the Design of Graphical Presentations of Relational Information. *ACM Trans. on Graphics, 5*, 2. April 1986. 110-141.
13. McCloud, S. *Understanding Comics*. Kitchen Sink Press. Northampton, MA. 1993.
14. Microsoft Corp., Microsoft Introduces V-Chat Communications for MSN, The Microsoft Network. Nov. 30, 1995. Microsoft Press Release. Redmond, WA 98052.
15. Montgomery R. and Gilligan, S. Comic Creator. Spark Interactive, 112 W. San Francisco St., Sante Fe, NM 87501, 1995.
16. Morningstar, C. and Farmer, F. R. The Lessons of Lucasfilm's Habitat. In *Cyberspace: First Steps*. Benedikt, M., ed. MIT Press, Cambridge, MA. 1991. 273-301.
17. Oikarinen, J. and Reed, D. Internet Relay Chat Protocol. Internet RFC #1459. May 1993.
18. Salisbury, M. P., Anderson, S. E., Barzel, R., and Salesin, D.H. Interactive Pen and Ink Illustration. Proceedings of SIGGRAPH '94 (Orlando, FL, July 24-29). In Computer Graphics Proceedings, Annual Conf. Series, ACM, New York. 1994. 101-108.
19. Seligmann, D. D., and Feiner, S. Automated Generation of Intent-Based 3D Illustrations. Proceedings of SIGGRAPH '91 (Las Vegas, Nevada, July 28 - Aug. 2). In *Computer Graphics, 25,* 4 (July 1991) ACM, New York. 1991, 123-132.
20. Trupin, J. The Visual Programmer Puts ActiveX Documents Through Their Paces. *Microsoft Systems Journal, 11*, 6. June 1996. 55-76.
21. Winkenbach, G., and Salesin, D. Computer Generated Pen and Ink Illustration. Proceedings of SIGGRAPH '94 (Orlando, FL, July 24-29). In Computer Graphics Proceedings, Annual Conf. Series, ACM, New York. 1994, 91-100.
22. Worlds, Inc. Worlds Chat: Meet Your New Cyberfriends. http://www.worlds.net/products/wchat/readme.html.