Sketch-Based Interface for Animation for Non-Experts

Daniel Wilches*, Pablo Figueroa*, Agustin Conde*, Faramarz Samavati†
*Department of Computer and Systems Engineering
Universidad de los Andes, Bogota, Colombia
Email: [d.wilches52, pfiguero]@uniandes.edu.co, condeagustin@hotmail.com
†Department of Computer Science
University of Calgary, Calgary, Canada
Email: samavati@ucalgary.ca

Abstract—We present a Sketch-Based Interface that allows non-expert users to create an animation from just pencil and paper. The interface works as a fast mockup tool for creating animations - turning the user’s freehand 2D sketches into 3D animations. To facilitate animation construction in offline or computer-scarce scenarios, special emphasis is placed on paper-based instead of tablet-based sketches. The interface makes use of stroke shape, proximity, and orientation to give the user’s drawings an animated interpretation. These Sketches use a 2D vocabulary of symbols, representing actors, actions and their relations. Custom symbols are defined by the user, and then associated with the relevant 3D assets within a database. The final composition of the sketches is passed into the system, employing image processing techniques for symbol recognition, and finally converting the user’s intentions into a 3D animation.

Index Terms—Sketch-Based Interfaces, content creation, 3D animation generation, 2D sketching.

I. INTRODUCTION

With an emphasis on education, not every school can afford ensuring each student access to their own computer, resulting in the sharing of one, or a few, computers within the class. This issue limits the activities teachers can plan for incorporating this modern technology. Consequently, it is important to focus children’s attention when working with technology, while recognizing that the children are unable to be in constant contact with the technology throughout the activity.

Considering this scenario, a technique to incorporate technology into students’ learning environments is presented. In this work, students can interact with computers using an interface ubiquitous in any classroom: pencil and paper. Here, children can draw stories on a sheet of paper and by scanning the sheet, can turn their drawing into an animation.

We propose a Sketch-Based Interface (SBI) that allows children to draw free-hand user-defined symbols to tell a story, and without prior knowledge of animation software, transform their drawings into animations. Due to the minimalistic input nature of this interface, simple drawings can be converted to interesting animations without understanding the complexities behind specialized software such as Blender [1] or Maya [2]. One sample of these simple sketches can be seen in Figure 1, and through analysis of this contour the system inflates the interface/system according to the users’ perceptions.

In section 2, the current approaches in SBI animation and modeling are presented and contrasted against our own approach. Section 3 presents the interface, its components, and intended uses and examples. Section 4 describes the pipeline developed to implement this interface. Section 5 presents the methodology and results of a user study of this work. Finally sections 6 and 7 conclude our work, presenting known limitations and areas for possible improvement.

II. RELATED WORK

We classify the work on SBIs in two main categories - directly associated with the work herein - SBI for Modeling and SBI for Animation. SBI for Modeling encounters similar problems to our own, namely the acquisition of the initial sketch and the recognition of user drawings and intentions. SBI for Animation is similar to our work for the need to get movement information from still-images resulting from the sketches.

A. SBI for Modeling

A variety of real time constructive systems have been developed to allow modeling through sketches. Igarashi et al [3] developed a system where the user draws a 2D contour, and through analysis of this contour the system inflates the

---

...
drawing into a 3D model; thereby constructing a 3D model from a 2D drawing. Since a 2D sketch is naturally ambiguous in 3D, such systems have complex heuristics for transforming the 2D into the additional dimension. Other authors ([4], [5], [6]) have also used sketches as the input for contour-driven reconstruction and deformation.

Karpenko et al [7] propose a system that is able to reconstruct a 3D model from 2D drawings. In their method the user draws some of the lines that correspond to the model, and the system infers other hidden contours and infers a smooth 3D-shape from them. In methods from other authors ([8], [9], [10], [11]) user-drawn strokes are applied to a 3D model surface and serve as handles for controlling the geometry. Users can easily add, remove or deform these control curves as if working with a 2D drawing directly. While those systems reconstruct 3D-models from detailed strokes, ours uses the strokes to retrieve 3D-models from a database. The 3D-models and the strokes to which they are associated don’t need to be similar in shape.

In the methods proposed by Lee and Funkhouser [12] and Yoon and Kuijper [13] users draw sketches to retrieve 3D models from a database in which each model has been previously associated with a sketch. Our work uses similar concepts to retrieve objects from a user database, but builds on these concepts allowing users to retrieve also animations and specify movement.

B. SBI for Animation

There are a diversity of batch systems that can turn a sketch into an animation. Davis et al [14] developed a system where an articulated human animation is generated in 3D from a sequence of sketches representing different keyframes in the animation. While in this system the user draws frame-by-frame the animations, in ours several animations had been pre-recorded and are available to be retrieved from a database according to the user’s drawing. This gives our method the advantage that the animation can contain more detail and their representation is exact.

Existing offline systems, such as the ones from Yi Lin [15] and Jain et al [16], allow the user to define 2D poses by editing a template skeleton and employing a motion capture database for pose generation. The former, given the input sketch, recommends possible 3D pose options from the database, while the latter makes a 2D projection from the motion capture data, aligns its timing and joint angles to match the input sketch.

Other important works are the real-time animation systems. In Motion Doodles [17] the user defines a route for a character by drawing a continuous sequence of lines, arcs and loops representing the location, timing and types of movement the character can perform. In Igarashi et al [18] the user deforms a 2D figure represented as a triangle mesh. Inside the mesh, the user defines and transforms control points, with the system repositioning the rest of the vertices while minimizing the distortion.

All these systems are focused in creating digital content around one character, ignoring other elements, such as the camera movements and actors in a story. Addressing these, our system complements previous works.

There are plenty of works that deal also with the processing of sketches, for instance the works of Pusch et al [19] and Olsen and Samavati [20] deal with the problem that most SBIs instruct users to employ single strokes where they may prefer to employ many small strokes for making more precise shapes. In those methods the strokes are analyzed and multiple strokes are replaced by a single stroke that reasonably estimates the user’s intention. Even though this beautification of sketches is useful in some cases, in our SBI the users don’t need to draw accurately the symbols, so high precision strokes are not necessary and neither this kind of sketch processing.

III. INTERFACE DESCRIPTION

The interface presented transforms a 2D-sketch seen from above, such as those in Figure 2 and Figure 1, into a 3D-animation, such as that in Figure 3. The example sketch of Figure 2 portrays a story of a robot and a truck -the actors- represented by user-defined symbols. The robot walks towards the truck, with the movement represented by an arrow, and the action represented by its name. In Figure 1 a more complex sketch can be observed. In this example, actors are jumping, walking or shooting, while moving or standing still amongst the environment of houses and trucks. Note that distinct symbols are carefully selected to ensure easy identification within our system. Even though, with a more careful, user centered, symbol design, the symbols could be chosen to be easily recognizable by users. One of the frames of the animation resulting from processing the sketch in Figure 1 into a 3D-animation is shown in Figure 3.

![Figure 3](image)

Fig. 2. Sample fragment of a 2D sketch that uses the proposed interface. The symbol to the left represents a robot, while the symbol to the right represents a truck. The arrow extending from the robot represents movement towards the truck. The word enclosed in the box indicates the action that the robot performs while moving.

The proposed SBI is based in easily-recognizable symbols, whose intention and use are as clear as possible to users. Symbols such as a stick-man could be used to identify a person, a dog’s silhouette to identify a dog, and a car’s silhouette to identify a car. Consequently, the user interface is intuitive enough to be grasped in a short period of time. The symbols representing the objects of the drawing are called actors. To represent movement in a scene, the user employs the intuitive symbol of an arrow indicating that an actor should perform a movement. Direction and curvature of arrows indicate the path actors follow while moving.
Fig. 3. One of the frames of the 3D-animation resulting from processing the image shown in Figure 1.

Fig. 4. Components of the interface. Actors are represented by user-defined symbols, and serve as the main building-blocks. Arrows extending from actors indicate movement, while the kind of animation the actor performs is specified by the Actions text.

A. Interface Components

As illustrated in Figure 4, the interface is composed by three different kinds of symbols: objects, which are interpreted by the system as the actors within the scene; arrows, which represent the path of movement by the actors; and actions, which are hand-written words indicating the name of the action to be performed. The next section details how the system processes these three classes of symbols.

1) Object Symbols: The interface’s objects are user-defined easily-drawable and more abstract symbols. Sample symbols and their corresponding 3D objects can be seen in Figure 5. The user defines and register the symbols by associating them with models from a user database. These symbols may be changed between stories, but it is recommended to follow these guidelines: (1) Symbols should be as simple as possible to make the drawing of stories easier and faster; (2) Symbols should be meaningful for the user, to reduce time spent looking up symbol associations; and, (3) Symbols must be sufficiently different from one another, to facilitate the symbol recognition process.

Symbols can be drawn anywhere in the sketch, and used as frequently as desired. Consequently, the amount of actors in a sketch is only limited by practical considerations like the size of paper, computer memory, etc.

2) Arrow Symbols: The interface supports arrows to allow specification of the movements of each actor. The object that is performing the movement is chosen as the closest symbol to the arrow’s tail. Since a scene is drawn from above, the direction of an arrow represents the X and Y components of movements. In the current system’s implementation the Z-component of the movement is always fixed to 0; so even tough the scene is being interpreted as a 3D-scene, all the movements and positions are constrained to a 2-dimensional space, and the objects lay on a plain ground.

3) Action Symbols: Describing an action with a simple sketch can become a hard task (e.g drawing a sketch for jumping). Using vocabulary for actions conflicts with the second rule above (reducing time for looking up symbol associations). However, using a simple word like "RUN", "JUMP", "WALK", etc, is easy and meaningful. For these reasons, actions are represented by a name, corresponding to predefined animations that an actor can perform.

In order to associate an action with an object within the scene, the name of the action must be enclosed within a box next to the object. Proximity is used to associate action with symbols in the scene.

The interface has been designed so that the user may add their own actions, and consequently configure their characteristics according to their needs.

IV. SYSTEM IMPLEMENTATION DETAILS

As illustrated in Figure 6, the pipeline for implementation consists of six tasks: symbol definition, sketch drawing, sketch acquisition, symbol recognition, scene interpretation and 3D scene generation. In the first task -symbol definition- the user provides a sample of the symbols that are going to be used through the pencil-and-paper drawing phase. In the second task -sketch drawing, or our pencil-and-paper stage-, the user employs the symbols to describe their story. In the third task -sketch acquisition-, users digitize the sketch made with pencil and paper, and create a digital image. In the fourth task -symbol recognition-, the system recognizes the symbols drawn by the user in task two against those defined in task one. Unrecognized symbols are then matched against the predefined set of symbols (arrows and letters). In the fifth task -scene
interpretation-, the system interprets the scene, giving meaning to the recognized symbols. Finally, the sixth task -animation generation- generates the animation, wherein the system takes the interpretation of the scene and converts it into a 3D-animation.

Details of each task are described in the following subsections.

A. Symbol Definition

Users can include custom and predefined symbols in their compositions. To support custom symbols, the user needs to draw them isolated, so the system is able to recognize them and have a clear silhouette of what the user wants to define as a symbol. Symbols created in this way represents the actors within the story. Even though the user is free to choose the symbols they want, the system requires that a symbol must be connected to facilitate recognition. This restriction is to support the algorithm used to split and tokenize the images into separate objects. Furthermore, the user-defined symbols must be unique so as to diminish ambiguity among them.

Once the custom symbols are created, the user proceeds to associate each one with existing 3D assets through a configuration file. If these assets contain animation data, the user can also specify which animation is going to be played when a given hand-drawn word accompanies the symbol.

On the other hand, pre-defined symbols in our system represent the camera, the rectangle that encloses a drawn word in a composition and the arrows that indicate movement paths.

B. Sketch Drawing in Paper

Within this stage, using pencil and paper, users create the compositions needed for visualizing the scene with the symbols available. Every composition is a sketch where the user draws any combination of the symbols, taking into account the following restrictions:

1) None of the symbols may overlap with others.
2) An actor must have at most one arrow and/or one handwritten word.
3) Handwritten words must be spelled in uppercase, and surrounded by a rectangle.
4) The composition must be drawn on plain white paper without grid lines or other designs.

After the user has constructed their drawing, it can be entered into the system to begin the process of converting to a 3D animation.

C. Sketch Acquisition

Once the drawing is made on paper it must be transformed by any means into a digital image and passed into the system. This transformation can be made, for instance, by scanning the paper or capturing it through a camera.

There are several sources of noise that must be addressed in order to have a clean image passed into the next phases. Common sources of noise are scanning and paper imperfections.

To deal with the noise the system applies a sequence of filters and removes small objects (15x15 pixels or less). To do this the system first scales the image’s histogram so too dark or too bright images can be better handled. Then it applies a mean filter on the original image, thus removing some of the noise. Finally it applies a threshold filter using the Otsu’s method. Any piece of noise that escapes this sequence of filters will be filtered-out by the pattern-recognition algorithm in a later phase of this process (unrecognized objects are ignored), thus not affecting the generated animation.

D. Symbol Recognition

After constructing a gray-scale, nearly noise-free image, the system proceeds to extract the connected components assigning a unique label to each one. The system then attempts to match each component against the user-defined and predefined symbols by using the Angle Quantization algorithm. If the component does not match any actor’s symbol, the system tries to identify it as a hand-written word, and if the match is unsuccessful, the system tries to find whether it is an arrow or not. This recognition task can be seen as a decision tree in Figure 7. These techniques are described in further detail within the following subsections.

The system is designed to allow the use of any pattern recognition algorithm that can work on images. For the purpose of this implementation Angle Quantization [21] is being used.
1) **Actors Recognition:** In order to recognize the actor symbols from the set of all components, the angle quantization technique proposed by Olsen et al [21] was implemented using $k = 16$ bins. Consequently, the one-pixel thinning and point tracker filters are applied before using angle quantization.

The fact that the element is divided in segments makes the angle quantization algorithm more accurate. As a result, the comparison metric (Euclidean distance) between two features becomes more precise. An experimental metric of 0.02 or less indicates that two features are similar. When comparing an element with every custom symbol, the minimum metric that meets the similarity criteria is the chosen one and the element is recognized as a candidate actor represented by the evaluated custom symbol. The dissimilar ones fall in the set of possible words and arrows.

2) **Words Recognition:** By leveraging the use of a rectangle that encloses any drawn word and recognizing that such a rectangle is detected in the actors set (with rectangles as predefined actors), the system may then apply a standard OCR. In this implementation Tesseract OCR [22] was used.

Having recognized an action’s name, the system checks if the actor closest to the surrounding rectangle has an action with that name. If it is successful, then the action is associated with the actor for the next phases.

3) **Arrows Recognition:** Every object detected neither as a word, nor an actor, is considered as a possible arrow. To determine if the object is indeed an arrow the system applies a thinning filter and then determines if the thinned-object is composed of three segments, three end-points and one intersection point. Any thinned-object meeting these criteria is recognized as an arrow and the longest segment is followed to determine the trajectory the arrow will represent.

**E. Scene Interpretation**

After having recognized all the symbols in the image, the system has then to interpret what the user’s intention was. For this, the system takes the information of the paths the actors must follow according to the arrows that are affecting them, takes the actions that represent the animations the actors must perform, and generate a script containing all this information. If the system-defined camera-object is present in the composition, the system determines whether a movement arrow has been associated with it, and if so, generates the path that the camera should follow when recording the scene.

For readability and loose coupling with an animation system, our current implementation generates an XML file describing the identified elements within the scene. This file contains all the symbols that were recognized by the system, the paths they follow and the actions they perform. This constructed file acts as the input for the next and final phase in the system’s pipeline.

**F. Animation Generation**

We developed an OGRE [23] application that reads the XML script file and generates a 3D-animation. The application uses information contained in the XML file to appropriately configure the camera, load the actor’s 3D models and their associated positions, trajectories and animations. Since a composition is drawn from a top perspective, all the elements are placed in the XY plane leaving the Z component at 0 (here Z representing the height).

All actors are orientated towards the camera, except for those actors that have movement trajectories, which are oriented towards their direction of movement. This is accomplished by calculating the angle between the current position and the next point in the path, and updating the actor’s direction. This calculation is performed along the line of movement, and is embedded within the XML file.

Camera movement is achieved by using the Animation framework provided by OGRE. Here, every point from the recognized trajectory is added to an animation track as a key frame on 0.05-second intervals. From this information, the system generates a smooth camera movement along the desired path.

**V. USER EVALUATION AND RESULTS**

Some tests were conducted to measure some aspects of the system such as recognition accuracy, identify symbols that are natural to users and acceptance of users to this kind of applications.

**A. Recognition Accuracy**

Normally pattern recognition methods learn after analyzing several, sometimes hundreds or thousands, examples of the same symbol. In our SBI it is necessary that the system recognize accurately the user intentions with just one example of each symbol.

To test this accuracy we interviewed a group of 40 people from a University environment, and asked them to choose 12
symbols of their own. They were asked then to draw each symbol in a sheet of paper, and next to each one to make four more examples, but this time they must vary orientation and size. In total each person drew 12 symbol templates and 48 repetitions. After that, we scanned the papers, divided the symbols, and, after filtering out invalid symbols, we consolidated in the database a total of 480 symbol templates and 1757 symbol repetitions.

To test the accuracy of the system we took each of the 1757 symbols repetitions and asked the system to which one of the templates were they more similar. The results showed that in 38.75% of the cases the system accurately selected the correct template.

A closer look at the results showed that symbols that contained more than three self-intersections, or that are formed of straight diagonal lines were usually not accurately recognized by the system, while symbols that contained circular strokes had the highest rate of recognition, 44%. It was also observed that several users drew similar shapes, so the system had in some cases templates among which the decision was difficult. This similarity in templates goes against one of the rules for symbol selection mentioned previously, so it has to be avoided for future uses of this SBI.

B. Expressivity of the Interface

In order to test if users were capable of drawing a story given this SBI we developed a test in a group of 28 people from a University environment. Users were instructed about the components of the interface and they saw an example of a story that used it. After that we told them a different story, and asked them to draw this new story following the interface’s guidelines.

There were 26 users who successfully employed the SBI to tell the story following strictly the guidelines. Two of them employed elements not present in the SBI components or decorated them by adding realism to the symbols. For instance, one of them added windows to the buildings and wheels to the cars, while the other added also windows to the building and onomatopoeias to express the actors’ feelings or story events.

From the results of this test we conclude that with the proposed interface it is possible to successfully tell a story, that it doesn’t involve any components that are hard to employ, and also, that we must encourage the choice of symbols with a user-oriented design.

C. Eligible Symbols

One of the premises of our system is to use symbols that are natural to users. To make a better selection of the symbols to include in the predefined symbol-set we interviewed a group of five children between ages four and five. Children were asked to draw a story without constraining the characters they could include (free-form drawing). After that, they had to tell their stories, so we can record what they were trying to draw with those symbols.

After the test was conducted and children had told their stories we found that children drew just a portion of what they were trying to tell, so most of their stories remained in their imagination, not on the paper. All children’s drawings contained a still-image of their story; thus lacking movement or timing information.

Among the symbols that they drew we could identify that children prefer drawing clouds, mountains, trees and the sun; but, those are not present in their stories, those symbols play just a decorative role in their drawings. Some other symbols that were found constantly in their drawings were animals, cars, houses and people; but, differently from the former, those symbols did play a role in their stories. For a sample see Figure 8. From this test we could identify that there are components that are entertaining for children, but are missing in our interface. Those are the non-story related symbol to specify how the environment of the stories should look.

We also noted that children use to draw things differently of what they think, their drawings not always resembled every aspect of their stories, and most of it remained in their imagination. This makes it impossible to generate an animation from just the sketches as the children use to draw, so they have to be instructed to embed more information while drawing.

D. Acceptance Study

We conducted a test on children to check their acceptance on SBI-based drawings and applications for drawing animated stories. The same group of five children who participated in the previous test were presented some story characters (see Figure 5). They were asked to draw a story using just those characters, but were instructed that they could use several times the same character. Then, children were asked to tell aloud their stories, so we could record what children were trying to express. After that, to test the children’s acceptance to such symbols, they were asked in which way they preferred to tell their stories -by using their own drawings, or by following the interface’s guidelines.

After the test was conducted, we could note that by presenting children with predefined characters they were constrained in what they could draw, and they didn’t try to be as expressive as in the previous test. For instance, in the previous test children decorated their stories with non-essential elements, while in this test four of the children drew only the four symbols. Just one of the children drew an extra element that was present in his story: a highway (see Figure 9).

When children were asked to tell their stories there was almost no difference between this test and the previous one: all the stories were full of elements not present in the drawings. One particular thing found in the drawings in this test was that two of the children tried to modify the interface’s symbols to make them more real or expressive. For instance, they added nose and eyes to the “robot” symbol, and wheels to the “truck” symbol. So we concluded that the symbols should be less abstract, trying to resemble more the objects they signify.

When children were asked about their preferences in drawing, two of the five told they definitely liked only free-form drawing style, not this kind of constrained drawing. The
remaining three children said they had liked this test as well as the former.

We also tested whether the children were responsive to developing stories with other technologies, so we presented Toontastic [24], an iPad tool for animated story telling. We let the children play with it for some minutes with the aid of one adult who gave instructions about its use. In this presentation children were responsive to the application, they were focused and wanted to use it. Nonetheless, we felt that the children were more amused with the colors and moving parts of the application than with the story-telling. It was also clear that such small children must be in company of an adult when using this technology, otherwise they get stuck and bore rapidly.

Another future direction of this work is to support trajectory, and more importantly, collision recognition. A final point of improvement is enabling the actors to perform their movements at different speeds, and to specify lists of actions that should be done at different points of the movement: at the beginning, while in the middle, or at the end.

**REFERENCES**


