Comics2D
Automatically Generating Comics
Tiago Alves
INESC-ID / IST
Av. Professor Cavaco Silva, Tagus Park
2780-990 Porto Salvo, Portugal
tiago.alves@tagus.ist.utl.pt

ABSTRACT
Comics is a well-known visual medium that is perfectly suited to tell stories and depict summaries in an effective way. Its strength comes from the combination of images and text. Computers have long been used to aid contemporary artists in drawing comics. However, having computers themselves drawing comics is another issue. Our goal is to create a novel framework that allows for the production of comics in a vast set of styles while depending on no external system. We decouple the comics creation system from the story and image sources, and provide a simple way to append new components that handle the comics elements in different styles. We believe that the possibility of automatically creating comics with different visual styles can allow for more accurate recreations of different stories and help the user better understand their significance.

Categories and Subject Descriptors

General Terms
Algorithms, Design, Languages.

Keywords
Comics, comics description language, comics generation, graphical stories.

1. INTRODUCTION
Comics are a form of visual art. Like an album of pictures that depicts all the moments in a trip, comic strips capture all the essence in a narrative and can tell us a story in a very effective way.

Only a few panels are enough to tell the story or pass the message that the artists intend. What makes this possible is the powerful combination of drawings and text. These properties make comics a privileged medium for the visual representation of summaries. A textual summary lacks the visual effectiveness and appeal, and a video summary is not always possible and can prove itself to make the seeking of important parts difficult.

There are at least two systems that have addressed automatic comics creation: Comic Chat [5] and Shamir et al.’s system [8]. However, both systems aren’t generic enough to be used outside their native environments because the story (characters, facts, events…) and the visual style are tightly coupled with the comics generation process. They lack a generic language to describe the story independently from the comics they generate and, at some extent, they also lack a way of changing the visual style of the comic strips.

Moreover, comics are made of many different components such as panels, characters or balloons and the automatic production of these components isn’t trivial. There are many unwritten rules that rely on the artist’s personal drawing style and creativity. Such aspects can influence the quality of the character representation (their current actions and thoughts, their placement and postures), the quality of the balloon representation (their placement and readability) and the reader’s overall perception of the story (the sequence of the narrative, the social relations between the characters and their underlying emotions).

Our aim is to create a comics generation system which is independent from the source of the story and is capable of producing accurate results with different visual styles. As a consequence we also intend to specify a comics description language that is capable of describing stories as comics while maintaining its essence.

In the next section we look at existent comics description languages and applications for automatic generation of comics. Section 3 depicts our own comics description language and presents the comics generation system. In Section 4 we describe the evaluation to which our system was subject and show some preliminary results. Finally, in Section 5 we draw some conclusions and outline future work.

2. RELATED WORK
When creating a comics generation system that is independent from the story source, it is required that the story is somehow passed on to it. Thus, we will analyse existing comics description languages [2] [3]. Further, in order to understand how comics can be created by computers, the known methods to automatically create comics are also going to be studied [5] [8].

2.1 Comics Description Languages
The Comic Book Markup Language (CBML) [2] and the Comics Markup Language (ComicsML) [3] are two languages that already exist to describe comics.

The ComicsML is a simple description language for representing comics electronically in a format-independent manner. It allows people to create and share comics by describing the information of each panel of the comic strip. This information includes a description of the actions and the text that the panel contains.
The other language, CBML, is an attempt to create a language that allows the digitalization of comic books with the intention of preserving them for a longer time. So, the objective of CBML is to develop a vocabulary and define complex metadata that can be used to describe every detail of a comic book. For instance, it is possible to store information about the publisher or the characters that are present in each panel. The text associated with both narrative text and speech balloons are also possible like in ComicsML. Complex and artistic layouts however can’t be easily described by CBML.

The CBML which focuses on comics from the traditional medium is more advanced than ComicsML which focuses on comics from the digital medium. What both have in common is that both are meant to describe an already existing comic be it with more or less detail.

2.2 Automatic Comics Generation

One way of automatically creating graphical content for comics is to take pre-drawn images, like characters and backgrounds, and join them to get a final image. We call this method automatic composition.

On the other hand, if we have access to the visual representation of what we want to display in the comic strip, we can simply take a screenshot and enrich its content. This method is called automatic transformation of dynamic graphics.

2.2.1 Automatic Composition

Comics Chat [5] is probably the best example of how to render a comic panel by taking individual images and laying them out together. The goal of this system is to depict online conversations occurring in a virtual chat room.

Every chat participant has a comic character that symbolizes him. Also, every comic character has a finite number of facial expressions and body poses that are independent. To determine the correct expression and pose for each situation, the system does a semantic analysis of the participant’s messages. If no emotion is found during this analysis, the system selects a neutral expression and pose.

Comics Chat also has rules for the inclusion and the positioning and orientation of the characters. Although it is recommended not to show every character in every panel, it is important to always show speaking characters. And those who are being spoken to should be at least in the first panel of a multi-panel dialogue. The position and orientation of the characters have to be in accordance to the occurring dialogue. This means that people who are talking to each other should face one another and should not have anyone between them.

In addition to the characters, the system has other pre-drawn elements to add richness and variety to the resulting panel, namely, backgrounds and other visual elements. These are chosen according to the semantic meaning of a phrase or the use of some keyword.

On the contrary, speech balloons are drawn dynamically. The algorithm is fairly simple but good enough to hide the fact that the balloons are computer-generated.

2.2.2 Automatic Transformation of Dynamic Graphics

Ariel Shamir et al. [8] presented a system that is able to represent dynamic graphics in comics art form. The goal of their system is to create a visual summary of a three-dimensional computer game in non-photorealistic rendering quality simulating the look and feel of comics.

The system consists of various parts, the first ones being the logger, the scanner and the director. These three subsystems together are responsible for the creation of the story that will be subsequently depicted. First there is the need to log the interactions happening in the game engine. These interactions are then transformed into a coherent story separated into scenes. Finally, the main interactions are selected by analyzing a function that returns the level of importance of each one of them.

At this stage it is still necessary to convert the 3D scenes into 2D images. This is achieved by the renderer, who is primarily responsible for setting up the camera. To accomplish this task, the director feeds the renderer with high level directives, such as the preferred type of shot, which include direction and zoom-factor. These directives are transformed by the renderer to explicit camera parameters. To choose the best shot, many shots are taken around the desired point. The one with the best visibility of the primary entity followed by the best visibility of the secondary entities is chosen.

Further, the renderer has to manipulate the newly created images so that they resemble hand-drawn cartoons. This is done in post processing by stylizing the images. Various image processing algorithms are applied to achieve the final result.

The creation of speech balloons is also the responsibility of the renderer. He creates simple balloons and splits them along consecutive panels if the text is too big to fit inside a single one. The balloon placement is rather simple, since the game characters seem to appear always at the centre of the screen.

Using this approach, the renderer needs to implement a camera planning algorithm to be able to take the required shots. Thus, camera planning assumes a vital importance in this type of comics generation.
3. THE COMICS GENERATION SYSTEM
Since our aim is for our comics creation system to be independent, we followed the automatic composition method as Comic Chat [5] did.

The comics generation system is comprised of three separate components. The first one is a comics description language that we defined. The second component is the system’s library where the images that will be used to compose the comic are stored. Finally, the main component is the application which transforms the incoming comics description into a visual representation in comics art form.

3.1 Comic Strip Description Language
The Comic Strip Description Language (CSDL) is an XML-based markup language that semantically describes the story that our system should visually depict. The organization of the document is set up like a comic strip.

The root XML node is the comic node where information, such as the comic title, can be defined.

Its children are all scene nodes. Their function is to separate the story into different scenes according to temporal and/or spatial distances or just change of subject in the story.

The children of the scene nodes are of the type panel or transition. The panel nodes describe what is caught inside a comics panel: an important moment or action of the story. The transition node is positioned between two consecutive panel nodes to define the panel-to-panel transition that happens at this specific position. It is basically an optional helper node to give additional information since the panel-to-panel transitions are an abstract concept that is implicitly contained in the already defined panels.

According to McCloud [6] there are six different transition types between panels:

- **Moment-to-moment.** This type of transition is used to break a single event into several parts, showing that little time has passed between them. This is useful to build up tension. They are hardly ever used.

- **Action-to-action.** These transitions focus on a single subject that is performing distinct actions. They are by far the most often used transitions.

- **Subject-to-subject.** Here, the reader is transported from one subject to another while staying within the same scene. His imagination is very important and it’s up to him to give some meaning to the transition.

- **Scene-to-scene.** Deductive reasoning is what’s required in this transition. They represent a change of scene, transporting the reader across significant distances of time and space.

- **Aspect-to-aspect.** When an artist wants to show different aspects of a place or idea while avoiding to show how time is passing, he chooses this type of transition. However, it is rarely used.

- **Non-sequitur.** This transition type is also seldom used. It offers no logical relationship between panels.

The panel node has three attributes. The first one is the border type, the second is the duration of the happenings depicted in the respective panel, and the third defines the panel’s importance. The two last attributes help the system to choose a technique to show how much time the panel portrays and if it should stand out from the other panels.

A single comic panel does not have to portray just a single instant in time as it can also represent an event that goes on for minutes.

Panel borders can have various functions. The basic function of panel borders is to simply limit a space in which objects and actions are drawn. Another use for borders is to not use them at all. This gives the frame the feeling of timelessness and unlimited space.
Inside the panel node we can find the nodes that describe the basic comics elements that make up a panel, namely, background, character, narrative, balloon, and soundEffect. In addition, the node camera is also defined at this level. Its purpose is to give information about the desired zoom-level and a target such as the face of a character or his pocket.

The background node defines the image that should be used to represent the panel’s scenery. Inside it we can optionally define one or more objects that are part of the scenery through the object node and the respective images that visually depict them. Moreover it is also possible to define a location of the background where the object should be placed.

Backgrounds are used to situate the reader by depicting the place where the events are occurring. Another use for the backgrounds is to evoke an emotional or sensual response in the user. Figure 8 explores this situation.

Further we can define what a character is doing through the action node and optionally what the target of his actions is. The character’s location can also be defined optionally and the node gaze defines where the character is looking at. Like in the background node objects can also be defined inside the character node. The difference is that the object’s location refers to a place on the containing character so as to inform that the character is this object’s owner.

If a given panel shall have narrative text the narrative node has to be used. This node makes it possible to define more than one narrative text for a single panel and optionally define a character or an object to which it is associated.

The balloon node specifies speech balloons. There are different types of balloons for different types of speech, namely, normal speech act, whisper, and thought, just to name a few. These balloons can then be combined to create melded balloons and connected balloons. Figure 10 exemplifies the mentioned balloon types.

The different types of speech are chosen with the type attribute. A melded balloon is achieved by specifying more than one phrase where each phrase node is the text of one balloon. Since a single balloon can represent the speech of more than one character, it is possible to define more than one owner node, which correspond to existing characters of the actual panel. In
case that this balloon needs to be connected to another balloon of the actual panel, the connection node can be used to indicate the target balloon.

A sound effect and optionally its origin can be specified by the soundEffect node.

Sound effects aren't actually real sounds, just a textual representation. So for example, if the artist wants to tell the reader that the phone in the image is ringing he can add the text “Riiiiing”. It is also usual to represent these words in a big font size and with some text effects to give the sound a greater importance.

![Image 1](https://via.placeholder.com/150)

Figure 11. Textual representation of the sound produced by the cutting of a vegetable. In McCloud [6].

It is important to note that the Comic Strip Description Language follows a loose typing approach. This means that, for instance, the possible emotions’ and objects’ names are not pre-defined. An object can be any name as long as there is an object in the system’s library with the same name.

The novelty of CSDL lies in its ability to describe comics in a way that it becomes possible to recreate them, at least to a certain degree. For instance, it is possible to specify which character or object was responsible for a certain sound effect. Or even what the current emotion and pose of a character is and where he is turned towards to.

### 3.2 Library

The library is where all the images that will be used in the comic are stored.

In the case of characters the necessary images are all the different facial expressions and body postures of each character. Separating the faces from the rest of the body allows for a more efficient reuse of a facial expression on a variety of body postures.

Objects on the other hand are single images. But they can also have multiple representations in the library to account for the different perspectives or positions in which the objects can be laid out.

The same principle applies for backgrounds. These images define the scenery in which the story takes place. To visually enrich the comic it is possible to store more than one image for the same scenery and so have slightly different perspectives.

Characters and backgrounds can have other information associated with them, namely, important positions on themselves. These positions are called “spots”. A spot on a background could be, for instance, a place on a depicted kitchen table. On a character a spot could specify the position of a pocket. This way it is possible for the renderer to know where a book object can be placed and where the camera has to point at if the respective nodes in the CSDL require it so.

The library is stored in XML format. A database might seem the more appropriate choice for this specific application but requiring a database management system to use the Comics2D application could be too troublesome.

To ease the process of creating a library, a subsystem helps the insertion of images and the specification of spots.

It is possible to have multiple libraries. So, for instance, we can have a library with the characters drawn in western style and another library with the same characters drawn in manga style. This allows for the same story to be depicted in two different styles.

### 3.3 Realizer

The realizer is the heart of the system. It consists of a plug-in based application.

The application’s main module has the responsibility of loading all the plug-ins and providing them an interface to access the library. After loading the plug-ins, the module loads the story description in Comic Strip Description Language format and feeds it to the plug-ins. This feeding process is done in seven phases, each one corresponding to the opening or closing of the key nodes of the CSDL (comic, scene, panel). So, for instance, in the first phase (PreComicProcess) every plug-in receives the full story description (the comic node and all its contents). The same goes for the last phase (PostComicProcess). The exception is the PanelProcess phase where the panel is actually produced. There the plug-ins receive the panel nodes of the comic story and produce the models of the visual representation of the story. Figure 12 shows the seven phases of the application loop and the CSDL nodes that trigger them.

![Image 2](https://via.placeholder.com/150)

Figure 12. Main loop of the Comics2D application.
Every plug-in is designed for a specific task. Generate speech balloons and handle the sound effects are just two examples. These plug-ins are only active in the PanelProcess phase. An example of a plug-in that is active in the last phase is a layout algorithm. This kind of plug-ins can just work when every comic panel has been computed.

For a comic to be minimally acceptable it needs to have panels with backgrounds, characters and balloons. Additionally, the created panels need to be laid out. The application provides four base plug-ins to handle these tasks but they can be easily substituted by other plug-ins.

The base plug-in for the speech balloons implements advanced algorithms to make certain that the results are pleasing. So, for the construction of the balloon itself it implements Hurst et al.’s [4] algorithm for adaptive layout. This allows the text to adapt to the elliptic shape of the balloon. For the placement of the balloon it implements Chun et al.’s [1] algorithm. It places speech balloons according to algorithms used in automatic cartography. So, speech balloons are placed relative to the belonging character while maintaining the correct reading order.

The system specifies models for the internal representation of each element of the final comic. The internal representation could have been left to the plug-ins to decide but that would hinder a plug-in from accessing the model of another plug-in as there would be no standardized models. And this is of vital importance in this application. For instance, the speech balloon plug-ins need to access the model of the character plug-ins to know where the characters have been placed in order to place the speech balloons accordingly.

When requested, every plug-in has to export its models of the various panels in a specific format. One example is SVG [7]. This usually happens at the end, when the layout plug-in requests all the created comics elements. This approach makes it easy for a comic to be presented as a traditional comic in a paged layout or as a webcomic on an infinite canvas.

4. RESULTS
In order to evaluate the proposed solution we carried out two distinct case studies. In the first one we tried to evaluate the ability of the Comics2D system as a generic comics generation system. In the second one we tested if our system is capable of functioning as a comics generation system for visual summaries of stories.

4.1 Comics2D as a Generic Comics Generation System
In the first case study we use Comics2D to recreate a comic strip originally by Comic Chat [5]. It was chosen because of its use of narrative text and close-up camera shots which made it interesting to recreate. The first step consisted in creating a library with the necessary images from the said system. Then, a CSDL description of a particular comic strip was written.

This particular case study was intended to evaluate if our implemented system properly addressed the problems of how to describe comics maintaining enough character-centred information to preserve the essence of the story, and how to generate comic strips with different visual styles based on such descriptions.

The main differences between the original comic and our recreation (see Figure 13) are the lack of camera settings and narrative text, and some issues with the speech balloon placement. Also, our result is obviously not exactly equal to the original comic due to the difference of the comics generation algorithms used by both systems. Nevertheless, providing the necessary plugins to create narrative text and handle camera specifications should approximate both representations even more.

By taking into account the referred comic strip and the one from Figure 14, we can see that our system is capable of producing comics with different visual styles. We have also shown that Comics2D can independently create comic strips of stories through different CSDL descriptions. We simply described Comic Chat’s original comic strip in CSDL and our system generated the corresponding result.

We conclude that Comics2D has proven to be an independent and generic comics generation system that is able to generate stories decoupled from the comics description and visual elements sources.

4.2 Comics2D as a Visual Summary Generation System
The second case study used Comics2D as a generation system for visual summaries in comics art form. For that, it relied on external systems which produce a particular story, summarize its events as a CSDL description, and passed that on to our system. This allowed evaluating if our implemented algorithms for the generation of the comics elements addressed the problem of how to correctly portray each element of the comic strip and so help the readers to better understand the depicted story.

The evaluation was done through an online survey and yielded exactly 100 responses. Our speech balloon’s quality evaluation showed that it can easily compare with human drawn balloons used in digital comics. Although this is a very good indication of how good our system constructs the speech balloons, we cannot conclude that our speech balloon is the best one. It is important to note that we have used one of the better speech balloons created by
Comics2D. Also, although we tried to use speech balloons that seemed us the better ones to compare with, there might have been better ones.

From the participant’s answers we can also conclude that the system library, in combination with the character creation module, correctly represents the characters’ emotional state and posture.

Our system also had a mechanism to allow subtle image variations amongst different body postures and backgrounds. The results of its evaluation were not very encouraging. Although the percentage of responders that preferred the not varied sequence is the lowest one, it is still too high to be acceptable. Moreover, the percentage of responders that did manifest no preference among the respective sequence pairs is also discouraging. We still think that Comics2D’s feature to lower the amount of repetitiveness is important. In longer comic strips it should, at least unconsciously, make the readers feel less tired of being constantly confronted with exactly the same backgrounds and character representations.

In general the responses were very positive. We think that the created algorithms and used techniques increase the overall quality and likeability of the generated comic strips.

The overall user satisfaction with the generated comics was not surprisingly high but we think it was positive enough. We also think that people understand that the system does not try to create art. Rather, it tries to exist as a viable solution to the fast creation of visual summaries of stories. This was confirmed by the responses to a question that tried to assess Comics2D’s suitability to work as a visual summary generation system in comics art form.

This allowed us to conclude that we have, at least partially, demonstrated that our implemented algorithms are accurate enough to help the comics readers understand the portrayed story through the correct representation of the different comics elements.

4.3 Example of a Generated Comic Strip
The image presented below is a result of the use of our system as a visual summary generation system. The portrayed events are present part of the summary of the original story that was then translated into CSDL and finally transformed into comics.

Figure 14. A segment of a comic strip created by Comics2D.

5. CONCLUSION
This paper presents a system that is able to automatically generate comics. The system is independent from the source of the story that it should depict with different visual styles and is capable of producing accurate results. The description of the comic is passed through the Comic Strip Description Language which is capable of describing stories as comics while maintaining its essence. Particularly, it stores character-centred information that allows the system to recreate the comic with the characters’ original emotion and postures.

The proposed framework which shows how versatile our plugin based architecture can be. It is possible to implement different techniques to achieve different results of every aspect of comics such as speech balloons and layout types.

The developed system underwent two distinct evaluations with the intent of testing the systems’ capacity to create different kinds of comics with different styles and to check the generated comics elements’ accuracy, respectively. For the first evaluation the results were very positive since our system was capable of recreating most of the original comic strip. For the second evaluation, the results were encouraging too since they mostly showed that our algorithms produced correct representations of the different comics elements.

We have so shown that it is indeed possible to create a generic comics generation system that has the capacity of creating comics with different visual styles that are accurate enough to be understood by a common reader.

6. ACKNOWLEDGMENTS
This work was partially supported by European Community (EC) and is currently funded by the eCIRCUS project IST–4-027656-STP. The authors are solely responsible for the content of this publication. It does not represent the opinion of the EC, and the EC is not responsible for any use that might be made of data appearing therein.

7. REFERENCES