Inverse Computer Graphics: Parametric *Comics* Creation from 3D Interaction

Ariel Shamir Michael Rubinstein Tomer Levinboim

Efi Arazi School of Computer Science, The Interdisciplinary Center, Herzliya

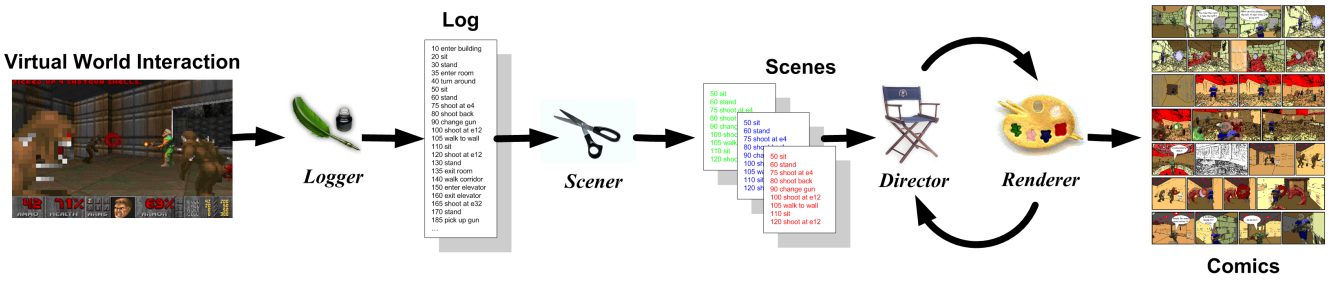


Figure 1: An overview of the process of transforming 3D graphic interaction into *comics* .

Abstract

There are times when Computer Graphics is required to be succinct and simple. Carefully chosen simplified and static images can portray a narration of a story as effectively as 3D photo-realistic continuous graphics. In this paper we present an automatic system which transforms continuous graphics originating from real 3D virtual-world interactions into a sequence of *comics* images. The system traces events during the interaction and then analyzes and breaks them into scenes. Based on user defined parameters of point-of-view and story granularity it chooses specific time-frames to create static images, renders them, and applies post-processing to reduce their cluttering. The system utilizes the same principal of intelligent reduction of details in both temporal and spatial domains for choosing important events and depicting them visually. The end result is a sequence of *comics* images which summarize the main happenings and present them in a coherent, concise and visually pleasing manner.

Keywords: Information Visualization, Non-Photorealistic Rendering, Comics, Succinct Graphics

1 Introduction

One of the major ongoing efforts in computer graphics is the strive for more elaborate, more complex and more realistic visual displays. This includes the strive for higher rendering quality and for higher frame rates. Nevertheless, there are times when more succinct use of graphics is required. For example, when there is no time to watch a whole movie, or to play a whole game; when it is too expensive to transfer or store elaborate graphic data, or when the target display device is crude or even static. In such situations, the graphic complexity must be reduced both in terms of quantity and in terms of quality, both temporally and spatially. Nevertheless, the need to carry a message or deliver information calls for intelligent techniques of reduction where significant information is preserved and unimportant data is dismissed. This paper investigates this type of graphics reduction, presenting a system for automatic creation of *comics* from a full scenario 3D virtual game.

We define this problem as *inverse graphics* since it uses as input the classic output of 3D graphics: we begin by obtaining a log of events originating from the interactions of players inside a 3D virtual world. This log is analyzed and processed based on parameters given by the user such as point-of-view and interest. Our goal is to create a sequence of *comics* images which summarize the main happenings and present them in a coherent, concise and visually pleasing manner. This transformation creates a more *succinct* representation of graphics by reducing both the temporal and the spatial complexity. Hence, the challenge is twofold:

1. In the temporal domain the challenge is to automatically detect and choose the most important or most interesting events in the interaction. The definition of interest depends on the point-of-view specified by the viewer.
2. In the spatial domain the challenge is to depict these events with one or more images that will convey their meaning faithfully. This includes choosing the points in time to portray an event, selecting camera parameters to create descriptive images, and choosing a rendering style to reduce their complexity while preserving their essence.

Screenplays, storyboards and scripts use comic-like frames to narrate a story. Inversely, such displays can be created to summarize the main events in a film, a video or a virtual world interaction [Brand 1997]. Furthermore, defining *comics* as *juxtaposed pictorial and other images in deliberate sequence* [McCloud 1994] covers not only artistic and entertainment uses, but other possible applications as well. For instance, in medical or scientific visualization juxtaposition can be used to compare or illustrate data. In education and training visual juxtaposition can be used for explanation of structures, assemblies or other processes [Agrawala et al. 2003]. In fact, the "small multiples" idiom used for envisioning any kind of information [Tufte 1990] can be interpreted as juxtaposed pictorial elements.

Traditionally in *comics* , selecting or producing the right image and text that communicate the right message is the work of an artist.

Indeed, we are still far from an artist's capability and expressiveness. However, this work goes one step towards this goal by presenting a system which is capable of extracting a sequence of important events from a continuous temporal storyline, and converting them into a graphical representation automatically. An overview of our system can be seen in Figure 1. While the user interacts with a game or moves through a virtual world, a *logger* 'listens' to all events and stores them in a log file (Section 4). The log file is cut into logical units or 'scenes' by the *scener* using a model for character interactions. Specific scenes and events are chosen according to the user definition of interest (Section 5). These are then converted to *comics* by an interaction between the *director* which chooses the frames and positions the camera (Section 6) and the *renderer* which is responsible for the style and abstraction of the images (Section 7).

The main contribution of this work is the creation of an automatic end-to-end system which transforms interactions in 3D graphics into a succinct representation depicted by *comics* . The system is capable of producing different *comics* sequences based on different semantic parameters provided by the user such as point-of-view and the level-of-details. To our knowledge it is first in attempting to create *comics* automatically, and use the same reduction principal in parallel both in the temporal and the spatial domains.

2 Related Work

Previous work has dealt with several aspects of our work. Analyzing a log of events is directly connected to story understanding which has been studied from early days of Artificial Intelligence [Charniak 1972; Schank and Abelson 1977], and is still an active research area [Rimmon-Kenan 2002; Mueller 2002]. The more thorough the understanding of the story, the better the selection of interesting and important events will be. Nevertheless, our approach does not depend on deep story understanding [Mueller 2004]. Instead, we use a model of the interactions between entities or characters in the world to recognize scenes and choose events.

Automatic creation of *comics* is also closely related to automatic creation of animations or movies [Karp and Feiner 1993; He et al. 1996; Amerson and Kime 2001; Halper and Masuch 2003; Friedman et al. 2004]. In both cases, a series of given events must be decomposed into scenes and visual display must be created. In fact, the general outline of our system follows that of similar systems for movie creation by first segmenting the events into scenes, then selecting specific scenes or scene parts, and then transforming each one into visual depictions. Moreover, some cinematographic rules must be followed also in *comics* sequences. For example, the 180° camera line rule applies also for *comics* images.

Two previous approaches have been suggested for the preservation of cinematographic constraints in movie creation. One is based on rule-based constraint satisfaction [Friedman and Feldman 2002] and the other uses idioms which are stereotypical ways to capture some specific actions as a series of shots [Christianson et al. 1996]. Our approach for transforming events into *comics* follows the later, but extends the notion of idioms to the creation of different types of *comics* frame transitions. Although video summarization [Brand 1997; Hanjalic et al. 1999; Fern et al. 2002] is also related to our work, most of the research in video has a somewhat different focus, which is image analysis and understanding. Our input comes from a 3D virtual world and its different nature includes more semantic information.

The language of *comics* is different from the language of cinematography. On the one hand it seems simpler since only static images need to be chosen. On the other hand, the choices are more critical and must convey movements or actions, which is more difficult to do using static images. Furthermore, the general approach of automatic movie creation is inherently photo-realistic. In contrast, we follow the same path of detail-reduction and abstraction in rendering as in the temporal story extraction. Since we do not have full geometric information for rendering, we create non photo-realistic images by applying post-processing stylization on the rendered images with the intension of enhancing important details and reducing cluttering.

Non-photo-realistic rendering (NPR) is an active field of research in computer graphics [Gooch and Gooch 2001]. A major trend in NPR is targeted towards the definition of stylized filters to achieve certain painting or sketching effects. In our work, we do not concentrate on artistic stylization but try and follow technique for reduction in details in order to enhance the essence of the image ([Herman and Duke 2001] used the term minimal-graphics). A similar abstraction approach is presented in [DeCarlo and Santella 2002], using a perceptual model of eye movements. We use the previous semantic knowledge from the virtual world and story extraction to distinguish foreground from background, main subject from superfluous details etc. Our stylization is based on two techniques from image processing: edge detection and clustering. For edge detection and enhancement we use anisotropic Laplacian operator [Trucco and Verri 1998], while the clustering and filtering are based on the k-means [Lloyd 1982] and mean-shift [Comaniciu and Meer 2002] algorithms.

3 The Language of *Comics*

Comics carry a somewhat childish reputation and are often dismissed as un-serious. Images in *comics* are often simplified and deliberately presented in a non photo-realistic manner. Still, *comics* are capable of invoking strong and emotional reactions, create identification and convey a story in an extremely effective and appealing manner. In fact, the use of symbolism and abstraction in *comics* can promote identification since realistic images resemble specific people or places whereas symbolic figures can represent anyone or anywhere. For these reasons *comics* narration can be leveraged to create effective and appealing succinct graphical depictions.

One of the primary principals of *comics* is the translation of time into space. The images in *comics* are inherently static, but they display dynamic information. The basic principal of depicting a dynamic events using a sequence of still images is so natural to human perception, that in most cases there is no need to explain it at all. Still, when considering how to create a sequence of still images to convey dynamic events, *comics* presents several possible transitions of time between frames [McCloud 1994]:

1. **Moment-to-moment:** breaks the action or motion into images based on time intervals. The results may look like a sequence of frames from a movie.
2. **Action-to-action:** breaks the action or motion into images based on the type or essence of the action happening. This transition is mostly used in *comics* .
3. **Subject-to-subject:** images switch from one action to another or from one character to another, but still remain within the same scene.
4. **Scene-to-scene:** consecutive images leap in time or space, usually signifying a change of scene.

Another leading principal in *comics* is the use of visual icons. The meaning of icon here is taken in its general sense as any image used to represent a person, place, thing or idea (see e.g. Figure 1). The use of abstract icons as opposed to realism focuses our attention through simplification, by eliminating superfluous features and creates higher identification and involvement.

We use these two principals to create a succinct depiction of 3D interaction. First by transforming a continuous sequence of events

into a discrete set of static images which portray them (Figure 4), and second by using abstraction in the visual depiction instead of realism.

4 Logging Events in a 3D World

The *logger* is responsible for tracing all events in an interaction and storing them in a log file. In order to apply our techniques in a real 3D-virtual interaction, the logger must be incorporated inside the engine of the 3D world. After examining several options we resolved into using a 3D game as the basis for the 3D interaction input. Not many open-source games allow modifications to the core engine (and it seems that this situation is not going to change anytime soon [Geitgey 2004]). For our experiments we use Doom [ZDOOM 2004], a simple 3D shooting game. Although we use the multi-player version of the game, the possible events and interactions in such a game are somewhat limited. Nevertheless, our system is general in a sense that new types of events and interactions can easily be defined, recognized and supported for any type of virtual interaction.

Each entity in the virtual world has a set of attributes which define its state. this can include its position, its possessions (e.g. tools or weapons), its current action and some given measures such as vitality or strength or hunger. An *event* in the virtual world is defined as any change-of-state of any of the entities in the world. Events are usually actions carried out by characters such as moving, picking up an object, sitting down, opening a door etc. Since many changes can happen simultaneously we measure the change-of-state of the *world* at some atomic time-unit called a *tick*. We store in the log all changes that occur to all entities between each two consecutive ticks. All of the event's related parameters are stored in the log such as the entity identifier, its positions, and the action type. The log also holds all the initial information regarding the geometry and entities of the world at time (tick) 0. Thus, at any tick t, we can reconstruct the state of the world based on the accumulated information from tick 0 to tick t. Some filtering out of 'unimportant' events is already done at this stage to prevent the log from exploding. Similarly, even though an action may last several ticks, only its outset is logged. This logging scheme was chosen since a large number of characters and events are involved in each scenario, and the log can grow considerably. Incremental logging needs significantly smaller storage compared to storing the full state-of-the-world at each tick.

5 Recognizing Scenes: The iHood Model

Following many narrational art forms such as theater and motion pictures, we decompose a story into a sequence of scenes. Hence, one of the first steps in transforming a log of events into a coherent story is to separate it into scenes. The *scener* is responsible for segmenting the log into scenes. The notion of a scene is difficult to define precisely. In general, a set of related actions or events that occur in one place at one time can be specified as a scene. Hence, the two main parameters defining a scene are usually time and space, and we can separate the events according to their location and time. Nevertheless, there may be many events occurring simultaneously at the same location, but only some of them are relevant to a specific story. Similarly, some events may occur in different locations but belong to the same scene (such as in a car chasing scene). Clearly, a simple model of time and space separation would not be sufficient, and some level of event understanding and classification should be performed.

Separating events into scenes also depends on the point-of-view. Before transforming a sequence of events into a story there is a need to select a point-of-view or narrator. story understanding is an extremely difficult problem, therefore, we approximate it with a simpler scheme for modeling interest or importance of events. We model the *interactions* between entities in the virtual world in order to recognize the scenes. We analyze interactions which involve two entities instead of examining events which are actions of a single entity. Furthermore, the interaction importance is factored by the importance of the entities and actions, enabling the creation of a bias toward a specific point-of-view or a specific type of events.

Figure 2: Two example of interaction neighborhoods (iHoods) of two entities the orange and the blue. The image shows a top view of a 3D game environments where each entity is represented by a circle. Entities are part of an iHood of another entity either if they are within its vicinity (large orange or blue circles) or they interact with it (dotted lines).

An *interaction* is defined as a relation between two entities in the world, for example 'see', 'shoot', 'pick', 'hold' can be interactions, but also spatial proximity can be defined as a feeble type of interaction. Let E be the set of all entities in the world, F the set of all possible interactions, and N the natural numbers (representing ticks). An interaction R is a function $R : N \times F \times E \times E \rightarrow \{0, 1\}$. If two entities e_1 and e_2 interact at tick t in an event of type f then $R(t, f, e_1, e_2) = 1$, else it is 0.

An *interaction-neighborhood* or iHood for short, of entity e is the set of all active interactions at tick t of entity e with any other entity in the world:

$$I(e, t) = \{R | \exists e' \in E, \exists f \in F, R(t, f, e, e') = 1\}$$

Each entity e in the virtual world owns an iHood which lists all entities it is currently interacting with and in what manner. These iHoods are updated each tick using two procedures. The first is continuously simulating all events of the log to find events that involve the entity e. The second is by examining the vicinity of entity e for the entrance or departure of other entities. The vicinity of an entity is a sphere S centered at the entity position and with a given radius r (see Figure 2).

For each interaction type $f \in F$ we define a weight w_f which is a real value number configured by the user. The weights denote the importance of these interactions in a specific story, e.g. from a specific point-of-view. Hence, different choices of weights for different interactions eventually govern if an action will appear in a story or not.

At any tick t we define the *level of interaction* of an entity e as:

$$l(e, t) = \sum_{e' \in E} \sum_{f \in F} w_f R(t, f, e, e') = \sum_{R(t,f,e,e') \in I(e,t)} w_f$$

For a specific entity e we can graph $l(e, t)$ as a function of the tick t (see Figure 3a). We find that the level of interaction has several peaks and valleys during the course of interaction. Furthermore, we can change the shape of this function by changing the weights of specific interaction types. Our key assumption is that high level of interaction reflects significance in the story and vice

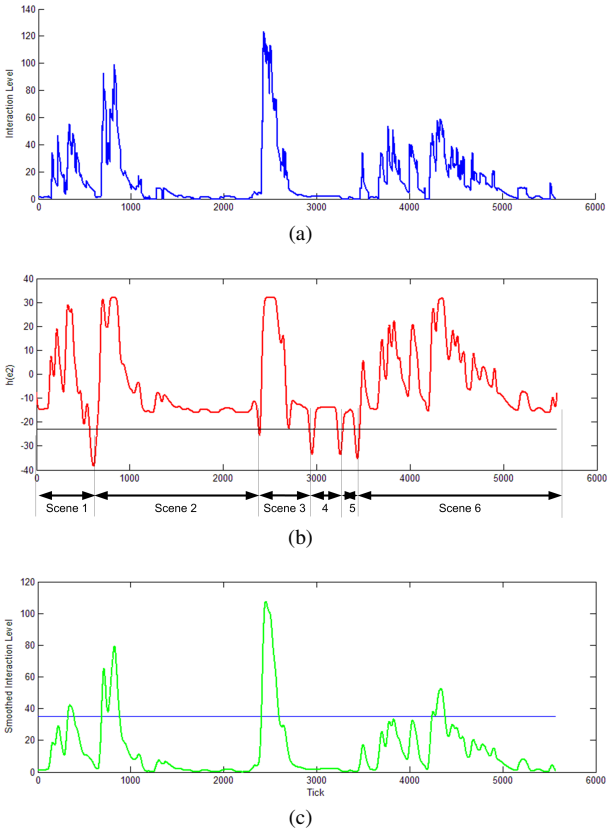


Figure 3: An example of the interaction level $l(e,t)$ over time of the sequence shown in Figure 11 is shown in (a). The smoother version $h(e,t)$ including also changes in location is shown in (b), and the scene partitioning based on it. Note, that scene 3 could have been separated to two scenes using slightly higher threshold. Choosing interesting scenes and events is based on a threshold on the smoothed $l(e,t)$ shown in (c). Note that scenes 4 and 5 would be cut-out since no interesting event occurred inside them.

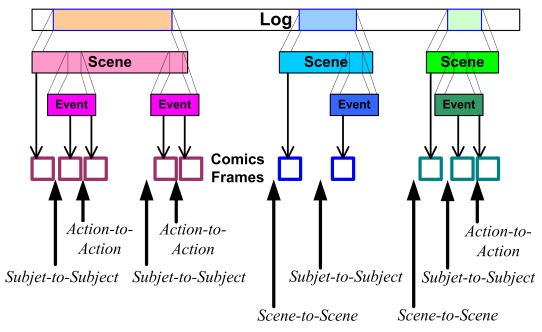


Figure 4: A schematic view of the visual conversion process from log to *comics* frames, and the places where different types of *comics* transitions are used.

verse. Hence, we would want to include the events around the peaks of $l(e,t)$ in any story involving entity e, and skip the events when $l(e,t)$ is low. This can be done using a simple threshold over the value of $l(e,t)$. Nevertheless, this simple scheme does not create a good partitioning of the story into scenes and is sensible to local fluctuations in $l(e,t)$. The interaction level $l(e,t)$ is an indication only for one of the parameters defining a scene - the time of the events. There is a needs to augment it with the second parameter - the location of the events.

We define a change-of-location function $s(e,t)$ which is 0 almost always apart from a finite number of times k where it is defined as a uniform spike with maximum height $c_k > 0$ ($s(e,t)$ is similar to a series of k approximated delta functions). The times when $s(e,t) \neq 0$ are the times the entity e has moved from one physical location to another and the constant c_k depends on the type of location change. This can include moving from one room to another, exiting or entering a building, or going up or down stairs. We subtract this function from the interaction level: $l(e,t) - s(e,t)$ since the separation of scenes should occur when both the interactions are low and there was a change of location. However, we also smooth the result using a gaussian $G()$ with local support to gain an ease-in and ease-out effect in scenes and events:

$$h(e,t) = G(t) * (l(e,t) - s(e,t))$$

$h(e,t)$ is a smoother version of $l(e,t)$ which also incorporates the change of location information (Figure 3b). We now use $h(e,t)$ to segment the log of events into scenes. The beginning of a scene is signified by a transition from a period of low $h(e,t)$ to a period of high $h(e,t)$ i.e. $\frac{\partial h(e,t)}{\partial t} > 0$, and conversely and end of scene is signified by $h(e,t)$ going down, i.e. $\frac{\partial h(e,t)}{\partial t} < 0$. Inside each scene, we still recognize interesting events by times when a smoothed version of $l(e,t) > T$ for some threshold T (Figure 3c). An *interesting-scene* would be one that has at least one interesting event. Non-interesting scenes, such as periods when the character is waiting, will not be shown since no interesting interaction has happened to the character, and hence $l(e,t)$ would be low all through this period.

Choosing a specific character as the main character and using this mechanism will result in a specific segmentation of the log into scenes. These scenes will represent a story from the point-of-view of this specific character. Changing the main character will create a different segmentation and a different story. Hence, this mechanism is a type of parametric temporal detail reduction, which is capable of emphasizing important information and removing insignificant data depending on the specific point-of-view.

6 Converting to Visual Depiction

Once the interesting scenes have been selected from the log file, there is a need to convert them into visual display. This is the main responsibility of the *director*. The director currently examines each scene independently, although inter-scene relations are certainly of importance and might be used in the future, for example, to change the order of scenes. Our goal is to transform each scene into a sequence of images which depict its main happenings visually (Figure 4). This transformation includes three major steps:

1. Choosing the specific events within the scene that should be portrayed.

2. Using one of the possible *comics* temporal transitions of frames (Section 3) to portray the events.

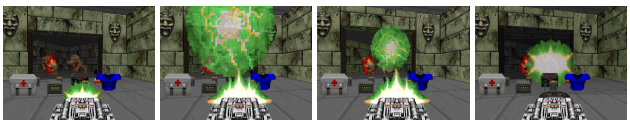
3. Choosing the camera parameters for each specific frame image: position, direction, zoom, etc.

Each scene is opened with one exposition frame which represents the scene-to-scene transition and creates continuity in the storyline. This frame is usually a wide angle image of the scene which gives the exposition of the scene. Next, the main events in the scene are recognized using a threshold on a smoothed version of $l(e,t)$ within the scene (Figure 3c). If more than one event is recognized, a subject-to-subject transition must be used. Nevertheless, by depicting each event separately, a subject-to-subject transition will emerge implicitly between the frames of the different events (Figure 5).

Each specific event can be displayed using moment-to-moment transition or action-to-action. Moment-to-moment transition is simple to create as the event is shown with k frames of equal intervals in time. Nevertheless, it is rarely used in *comics* and mainly to stress very important actions or to build tension by prolonging the perception of time (Figure 5(d)). A more appropriate and useful type of transition is the action-to-action transition. This transition depicts an event using fewer frames presenting significant movements in the event. Although even the same action performed by the same entity on different occasions may differ in the choice of significant frames, in most cases the frames depend on the type of action and its interval in time. Therefore, to depict an event using an action-to-action transition of frames we use pre-defined idioms depending on the type of event.

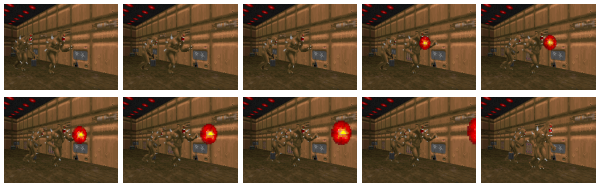
In general, an *idiom* is a mapping between a specific pattern of events in the scene to a specific sequence of time-frames. We can use the same idiom for different types of events, or, depending on the state of entities, choose different idioms to depict the same event. For instance, for a firing sequence the director uses an idiom which displays two frames in an action-to-action transition manner: one just before the peak of the action (the peak in $l(e,t)$) and one a little after the outcome (see Figure 5). A single peak in $l(e,t)$ containing interactions with many entities, is depicted by an idiom of a single wide-shot frame (such as frame 10 in Figure 11). Another example is when a player is shielded. In this case, it is important to stress its personal point-of-view and the director uses an idiom which defines a first-person point of view shots (frames 12,15 in Figure 11). Idioms can also define higher level mappings between otherwise individual events. In a conversation scene many single events of interactions are replaced by one prolonged idiom which later enables the insertion of text-balloons (Figure 9). Although in this specific game the types of interactions are limited, the idiom mechanism is versatile and easily extendible to other fields including other types of interactions.

(a) A firing sequence of the green player is depicted using an idiom composed of two action-to-action transitions and one subject-to-subject.



(b) The view of the snapshots from (a) as seen in the game, without using our system cameras.

(c) The same type of idiom depicting a similar event from the blue player's story, only this time the player is getting hit.



(d) An example of a moment-to-moment transition, which is not used in practice.

Figure 5: For firing sequences the director uses an idiom which is composed of two sets of action-to-action transitions: one for the firing entity and one for the entity being hit. This creates a subject-to-subject transition implicitly between the two sets.

7 Rendering: Visual Abstractions

As a result of the director's process, the happenings of a specific player or entity in the virtual world are portrayed using discrete points in time defined by idioms or the times when the interaction level is high. However, there is still a need to convert the 3D scene at these points in times into 2D images. This is done by the *renderer*, which is responsible for positioning cameras and choosing its parameters (Figure 6).

The director uses idioms which define high level directives. This includes the main entity in the scene, or a point in space at the center of the scene, and a list of secondary entities, if they exist, that should be portrayed if possible. It can also include the desired type of shot: close-up, medium, long, or wide shot, the height of camera, the roll, pitch, and yaw of the camera etc., although these last parameters are not used in practice. All these directives are translated by the renderer to explicit camera parameters. For example, a central point or entity is translated into a direction and the type of shot to a distance, creating a circle around the point of possible camera positions pointing towards the center. A number of camera positions are sampled around the circle uniformly and shots are taken. If there is an obstacle such as a wall or a pillar, the camera is advanced towards the center until it is visible. Factoring in other constraints (such as the 180° rule) eliminates some of the images. The remaining images are weighted first based on the visibility of the primary entity and then based on secondary entities, and the best one is chosen (Figure 7).

To create the actual *comics* images, the style of shading follows the same principal as the temporal selection: intelligent reduction in details. There are two options for achieving a certain style while rendering 3D graphics. The first is to use a specific shading style during rendering, and the second is to apply post-processing on the images. There are several 3D rendering engines and techniques which give the objects a cartoon-like form. Unfortunately in our case, the game engine supports only the original gloomy type of shading of DOOM. Furthermore, the game itself has very low quality graphics and is based primarily on sprites and textures. Therefore, instead of full 3D data, we can acquire only fixed resolution $2\frac{1}{2}D$ data from the game. We use the original game rendering-style image, extract the foreground and background masks, and use the texture stencils to gain 2D information on the background geometry. Using all these we stylize the image in post processing. Our goal for the visual end-results is to create the look-an-feel of *comics* , reduce the cluttering of the original images, and enhance the main characters and events in the images.

Several image processing techniques are combined to achieve the desired results. Color clustering based on mean-shift is used for the background and k-means clustering for the foreground. Edge enhancement is achieved by applying an anisotropic Laplacian kernel

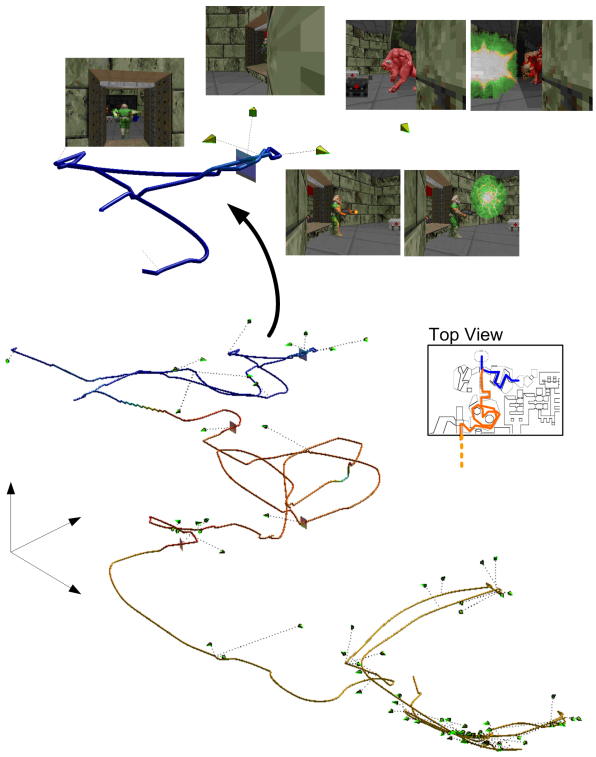


Figure 6: A plot of the full path of the green player during the interaction. At different points in time and space, and based on the interaction level, the director chooses a snapshot. In this example, the granularity is high and many shots were taken. The renderer is responsible for positioning cameras (green pyramids) to shoot the *comics* frame. Some examples of images taken from different cameras are shown closer-up at the top. Some cameras take two shots from the same position, for example, to create an action-to-action transition.

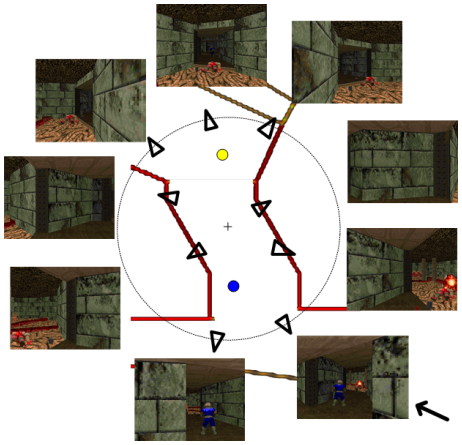


Figure 7: For each specific point in time 18 camera positions are sampled around the event's central point (only 9 are shown). In this top view of a scene, the point is between the two main entities - the blue player and a yellow monster. The best resulting image from the 18 is chosen for use in the *comics* . In this case it is the lower right image.

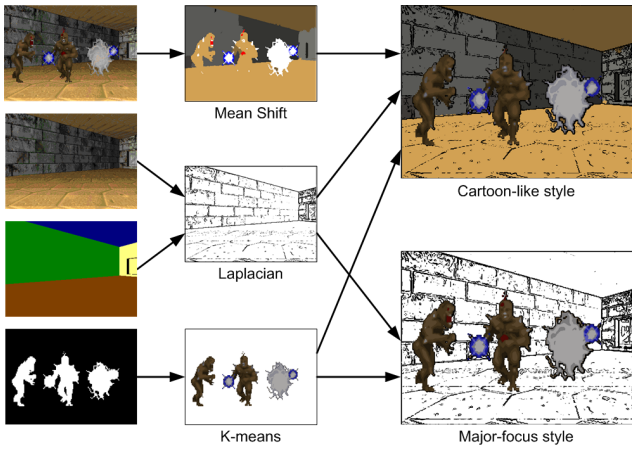


Figure 8: The $2\frac{1}{2}D$ image data extracted from the game engine and the different image processing techniques used to create the image stylization for our *comics* . There are two different styles, one is cartoon-like and one enhances the entities in the foreground by removing all color from the background.

on the image creating stroke-like edges. Combining all these together creates the resulting images (Figure 8). The renderer is also responsible for creating text balloons when a conversation takes place in the log. Currently, the idiom used for conversation is a simple alteration of images in front view of the speakers. Depending on the length of each image text, the renderer chooses the size of the balloon, and positions it to the left or right of the head, alternating between the speakers. The position takes into account the foreground mask of the speaker, the masks of other entities and the borders of the image. If the text does not fit into one balloon, it is distributed into several consecutive equivalent images (Figure 9).

8 *Comics* Layout

The last stage in the *comics* creation is the page-layout stage. After rendering and stylization, all images are in the same size and can be displayed consecutively one next to the other similar to the layout of Figure 9. Nevertheless, many *comics* pages do not look like a sequence of uniform images, often they present variations in the size, aspect ratio and positioning of the images to create effects such as stressing or prolonging. Although imitating the full skills of a *comics* artist is still a desire, we devise a layout algorithm for images that breaks the symmetry and creates the look-and-feel of *comics* .

To simplify the problem, we constrain all rows in our *comics* to be in the same height, which is often true in many real *comics* sequences. This means that our layout algorithm is transformed into a one dimensional problem of choosing the horizontal length of images in each row. Next, we separate the images to four basic classes based on semantics, e.g. the idiom or event they portray. Type B (big) images are those that can be expanded. These include images where maximum interaction of many entities is depicted, or exposition images of a new scene. Type S (small) images are those that should be condensed. These are usually a part of a sequence of images such as an action-to-action sequence. Type F (fixed) images are those that must remain in fixed size. In our sequences any image that includes text balloons is defined as fixed to protect the balloons layout. Any other image is defined as N (neutral) and can be expanded or condensed.

The layout process takes as input the fixed-length of all rows. We always choose a whole multiple of the length of a regular image

Figure 9: An example of the output of the layout algorithm for placing text balloons in our *comics* . The direction of the balloons alternate between entities and the text is broken if it exceeds a certain size.

(usually k=4). The layout proceeds in a greedy manner by fitting sets of up-to k images, one row at a time beginning from the first image until the last. At the beginning of each row it examines the next k images. If this set does not includes any B-type image than k images are put in the row with fixed size. Otherwise, the first B-type image is put aside as a 'filler'-image. For all other images, the layout tries to comply their constraints based on their types. Every B-type image is assigned a random expansion ratio between 1.2 and 2.0. Every S-type image is assigned a random condensing ratio between 0.7 and 1.0. Lastly, the 'filler'-image is expanded to compensate for all other images. If this is not sufficient, other N-type images are expanded. If no solution is found, the layout tries another round with different random values. If after a few tries it still fails, it inserts the images in their original sizes and proceeds to the next row. This basic algorithm is further enhanced for specific situations, for instance, when two consecutive B-type images are found and more.

Once the expansion and condensing factors are set for a row of images, the images are pruned at the fringes either horizontally (mostly for expansion) or vertically (mostly for squeezing) to achieve the right aspect ratio. Pruning is done in a symmetric manner from the top and bottom, or left and right, respectively with as little damage to the foreground as possible.

9 Results

We present results created from 3D world interactions of a two-player episode of the DOOM game. We denote the two players as the 'green' player and the 'blue' player. The original interaction takes 160 seconds and can be seen from the green player's point of view in the attached video (see regularInteraction.avi). All examples were created on an Intel pentium-4 2.8GHZ 500MB.

The log created from the interaction is 6.3 MB large in XML format and includes 5574 ticks. The scener first runs on the log to compose an interaction-script which includes all the iHoods for all entities. This takes 392 seconds, creating a script size of 683KB. This enables subsequent calls to the scener to be optimized and take only a few seconds. Hence, given a specific entity (point-of-view) and a threshold for scene cutting (granularity), the scener uses the interaction-script and log to create a scened-log. This takes 3 seconds for the blue and green players. The log is larger in size (8.68MB) since it is now segmented into scenes, and at the start of each scene it contains a synchronization point for all entities in the world. Later, this enables each scene to be portrayed independently and may be used to change the order of scenes in the future.

The director uses the level-of-detail threshold and converts the scened-log into a directing-script based on this threshold. This script includes the high-level directives used for creating snapshots of the story over time. In our example it is 60.4KB large. We used several different thresholds to create several level-of-details of the story for both the green and the blue players (see attached HTML pages).

Next, the directing script is used to create the *comics* images. Since we do not hold the full geometry, we use the game engine itself to render the basic images, and extract all the masks for stylization. Stylization is done by filtering all images in a post-process. The filtering of each image is the most expensive task and currently takes between 90 seconds for focused-style (Black and White background) to 150 seconds for Cartoon-style images.

Assuming E is the number of entities in the world, T is the number of ticks and V is the number of events, the most costly action in the process beside rendering is the update and creation of the iHood for each entity which is $O(T \cdot E^2 \cdot V)$. However, the rendering stylization process is in fact the major bottleneck since its complexity is dependent on the number of pixels and not entities or timesteps.

In Figures 11 we present the results of the green player story using the largest granularity. This means only highlights of the interaction are shown. Lowering the threshold will yield more frames automatically as can be seen in the attached HTML files. We used four different level of details for each of the two players: the blue and the green players. We also present all these *comics* in the second rendering style and an example can be seen in Figure 10.

10 Conclusion

This paper describes an end-to-end system for the automatic creation of *comics* from 3D graphics interaction. The major challenge met by this system is the transformation of elaborate and continuous graphic data into a discrete and succinct representation. In both the temporal and the visual domains we follow the same principal of abstraction by reducing details, focusing on major events and main characters. The system is built on top of a real 3D game engine and is able to trace and log the happenings in a multi-user world, and transform them into *comics* automatically.

There are many possible extension for this work and many enhancements need to be addressed. In terms of story understanding,

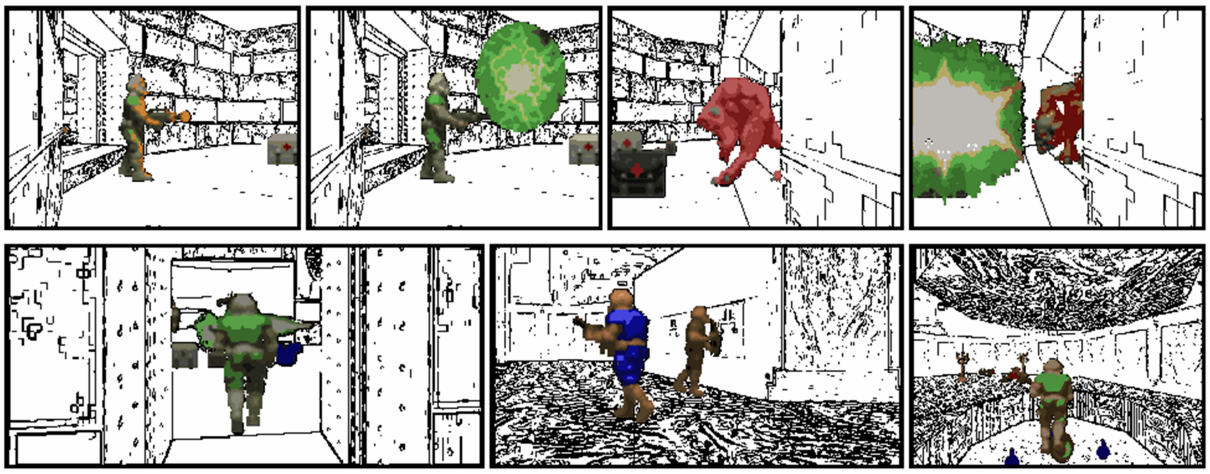


Figure 10: An extract from the green player *Comics* sequence created in the rendering style stressing the foreground. A larger portion with the cartoon-style rendering can be seen in Figure 11.

causality between events can be utilized to create better scene partitioning and to identify important events. In terms of directing, the current system for choosing transitions based on idioms is almost memoryless. Remembering the history of frames and idioms used may create a smoother and more interesting flow in the story. In terms of rendering and stylization, the use of 3D rendering instead of 2D image processing will open new possibilities. Lastly, in terms of layout, algorithms for combining text and images automatically to create effective displays, and algorithms for automatic page layout are still in their infancy and should be investigated further.

As mentioned earlier, the creation of *comics* and movies have much in common. In fact, the current system can also support the creation of simple movies instead of *comics* frames. Nevertheless, additional research is needed for the creation of camera movements and dynamic flow of display to support the automatic creation of high-quality movies as well as better *comics* .

References

AGRAWALA, M., PHAN, D., HEISER, J., HAYMAKER, J., KLINGNER, J., HANRAHAN, P., AND TVERSKY, B. 2003. Designing effective step-by-step assembly instructions. *ACM Transactions on Graphics 22*, 3, 828–837.

AMERSON, D., AND KIME, S. 2001. Real time cinematic camera control for interactive narratives. In *proceedings AAAI SSS 2001*.

BRAND, M. 1997. The "inverse hollywood problem": From video to scripts and storyboards via causal analysis. In *Proceedings 14th Conference on Artificial Intelligence*, 132–137.

CHARNIAK, E. 1972. Toward a model of children's story comprehension. Tech. Rep. AITR-266, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

CHRISTIANSON, D. B., ANDERSON, S. E., HE, L.-W., SALESIN, D. H., WELD, D. S., AND COHEN, M. F. 1996. Declarative camera control for automatic cinematography. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 148–155.

COMANICIU, D., AND MEER, P. 2002. Mean shift: A robust approach towards feature space analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence 24* (May), 603–619.

DECARLO, D., AND SANTELLA, A. 2002. Stylization and abstraction of photographs. In *Procceedings of ACM SIGGRAPH 2002*, 769–776.

FERN, A., SISKIND, J. M., AND GIVAN, R. 2002. Learning temporal, relational, force-dynamic event definitions from video. In *Eighteenth national conference on Artificial intelligence*, American Association for Artificial Intelligence, 159–166.

FRIEDMAN, D., AND FELDMAN, Y. 2002. Knowledge-based formalization of cinematic expression and its application to animation. In *Proc. Eurographics 2002*, 163–168.

FRIEDMAN, D., FELDMAN, Y., SHAMIR, A., AND DAGAN, Z. 2004. Automated creation of movie summaries in interactive virtual environments. In *Proceedings of IEEE Virtual Reality 2004*, 191–199.

GEITGEY, A. 2004. Where are the good open source games? *OSNews http://www.osnews.com/story.php?news_id=8146*.

GOOCH, B., AND GOOCH, A. A. 2001. *Non-Photorealistic Rendering*. A K Peters.

HALPER, N., AND MASUCH, M. 2003. Action summary for computer games – extracting and capturing action for spectator modes and summaries. In *Proceedings of 2nd International Conference on Application and Development of Computer Games*, 124–132.

HANJALIC, A., LAGENDIJK, R. L., AND BIEMOND, J. 1999. Automatically segmenting movies into logical story units. In *Visual Information and Information Systems*, 229–236.

HE, L., COHEN, M. F., AND SALESIN, D. H. 1996. The virtual cinematographer: A paradigm for automatic real-time camera control and directing. *Computer Graphics 30*, Annual Conference Series, Siggraph 96, 217–224.

HERMAN, I., AND DUKE, D. J. 2001. Minimal graphics. *IEEE Computer Graphics and Applications 21*, 6, 18–21.

KARP, P., AND FEINER, S. 1993. Automated presentation planning of animation using task decomposition with heuristic reasoning. In *Proceedings of Graphics Interfaces 93*, 118–127.

LLOYD, S. 1982. Least square quantization in pcm. *IEEE Transactions on Information Theory 28*, 129–137.

MCCLOUD, S. 1994. *Understanding Comics: The Invisible Art*. Harper Perennial.

MUELLER, E. T. 2002. Story understanding. In *Encyclopedia of Cognitive Science*, L. Nadel, Ed. London: Nature Publishing Group.

MUELLER, E. T. 2004. Understanding script-based stories using commonsense reasoning. *Cognitive Systems Research 5*, 4, 307–340.

RIMMON-KENAN, S. 2002. *Narrative Fiction: Contemporary Poetics*. Routledge, London, 2nd edition.

SCHANK, R. C., AND ABELSON, R. P. 1977. *Scripts, Plans, Goals, and Understanding: An Inquiry into Human Knowledge Structures*. Lawrence Erlbaum, Hillsdale, NJ.

TRUCCO, E., AND VERRI, A. 1998. *Introductory Techniques for 3-D Computer Vision*, 1st ed. Prentice Hall.

TUFTE, E. 1990. *Envisioning information*. Graphics Press.

ZDOOM, 2004. http://zdoom.org.

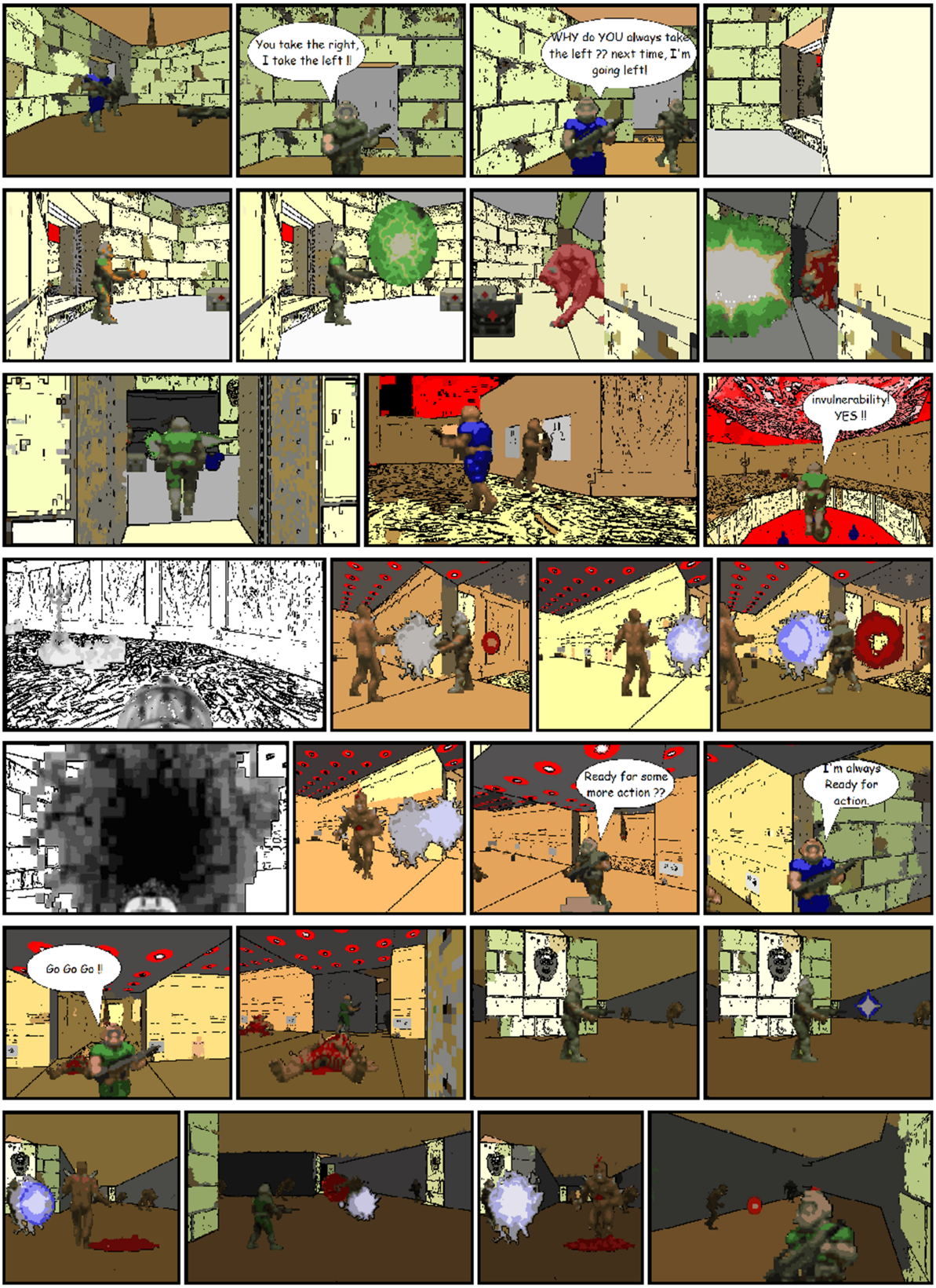


Figure 11: The beginning of the *Comics* page from the point of view of the green player. The full sequence and more examples can be found in the attached HTML materials.