AUTOMATING THE CREATION OF 3D ANIMATION FROM ANNOTATED FICTION TEXT

Kevin Glass and Shaun Bangay
Department of Computer Science
Rhodes University
Grahamstown, South Africa

ABSTRACT

This paper describes a strategy for automatically converting fiction text into 3D animations. It assumes the existence of fiction text annotated with avatar, object, setting, transition and relation annotations, and presents a transformation process that converts annotated text into quantified constraint systems, the solutions to which are used in the population of 3D environments. Constraint solutions are valid over temporal intervals, ensuring that consistent dynamic behaviour is produced. A substantial level of automation is achieved, while providing opportunities for creative manual intervention in animation process. The process is demonstrated using annotated examples drawn from popular fiction text that are converted into animation sequences, confirming that the desired results can be achieved with only high-level human direction.

KEYWORDS

Text to scene conversion, story visualisation, computer graphics, constraint optimization

1. INTRODUCTION

The conversion of fiction text to a unified multi-modal animation is a difficult and subjective process which requires substantial human intervention at every step of the process (as evidenced by the lengthy list of credits at the end of any film). Some steps in the process stand to benefit from automation, freeing the human component to concentrate on the creative aspects. This paper describes a process for automating tedious aspects of converting fiction text to animations, while still allowing directorial intervention consistent with the constraints in the scene.

We transform annotated fiction text into 3D animations by determining time-quantified constraints with regards to objects in the scenes, the solutions to which are used to specify layout, appearance and motion within a 3D environment.

Text is annotated according to a small set of categories as listed in Table 1. Each category is chosen and defined in a manner such that automatic information extraction processes may be used for the creation of annotations over fiction text (Glass and Bangay, 2005, 2006). The format of the annotations supports additional human manipulation to further refine the animation. The remainder of this paper assumes the existence of such annotations.

Table 1. Summary of annotation categories

Category	Fields	Description
Avatar	-	Explicit identification of a character
Object	-	Explicit identification of an object
Setting	-	Explicit identification of the environment
Relation	*type, subject*	Explicit description of a spatial relation
Transition	*type, subject, object*	Explicit indication of entry to or exit from the scene

They had it on the top of a hill, in a sloping field that looked down into a sunny **<setting>valley</setting>**. **<avatar>Anne</avatar>** didn't very much like a big brown **<object>cow</object>** who <**transition type='ARRIVE' subject='cow'>came</transition>** up **<relation type='near' subject='cow' object='her'>close<relation>** and stared at her, but it **<transition type='DEPART' subject='it'>went</transition>** away when **<avatar>Daddy</avatar>** told it to.

Figure 1. Example of annotated fiction text, sourced from *The Famous Five: Five on a Treasure Island* by Enid Blyton

Figure 1 presents an example of annotated fiction text. Fiction writing tends assume much implicit knowledge, and therefore any details not explicitly mentioned are not automatically annotated. As a result, there is an opportunity for manual intervention in the process during which human creativity may be used to insert further annotations into a scene.

After a survey of previous approaches to text-to-scene conversion (Section 2), we describe the steps used in our approach. The first step in generating 3D animations from annotated fiction text is the automatic translation of annotations into abstract constraints (Section 3), from which a set of quantified constraints is derived and solved using interval methods (Section 4). The scene specification is used to instantiate a 3D virtual environment, which is rendered into a final animation (Section 5). We present and assess examples of automatically produced animations in Section 6.

2. RELATED WORK

Of the existing research available in the field of Text-to-Scene conversion, three notable systems exist that produce animations from natural language input. The SWAN system converts stories expressed in natural language to 3D animations, but requires that input be expressed in simplified Chinese language (Lu and Zhang, 2002). The CARSIM system converts road accident reports into corresponding 3D animations, but is restricted to the car accident domain (Johansson et al., 2005). CARSIM and SWAN are notable in the fact that the exposition of the natural language input is in *story* form, that is, consisting of a sequence of descriptions and events, but neither of these systems use text sourced from popular fiction. The CONFUCIUS system, however, is capable of converting single sentences, possibly sourced from fiction books into corresponding animations (Ma, 2006). Both SWAN and CONFUCIUS make use of intricate knowledge-bases that provide the ability for reasoning with regards to the automatic interpretation of text and the layout of a scene. A result of this is the ability to generate highly detailed animations, but require some restriction in the form of input as well as substantial effort in the creation of the knowledge-base. We present an alternative method in which the degree of world knowledge required for the automatic conversion process is kept to a minimum, rather leaving this aspect to human creativity.

Other systems exist for producing 3D graphics from natural language sources, but none include the aspect of time nor use fiction text as input. Such systems include the 3DSV project (Zeng et al., 2003, 2005) and WordsEye (Coyne and Sproat, 2001). Other natural language visualisation approaches exist, including the use of images (Joshi et al., 2004; Glass et al., 2007), and the general use of natural language as an interface for graphical systems (Clay and Wilhelms, 1996; Badler et al., 2000).

Our work is the first to transform text sourced from popular fiction into corresponding 3D animations without prior language simplification. This research proposes a new phrasing for input of the Text-to-Scene conversion problem, particularly transforming *annotated fiction* text into animations, as opposed to conventional techniques that create a number of intermediate semantic representations (for example, predicate argument structures or semantic frames (Coyne and Sproat, 2001)) from natural language.

3. CONVERTING ANNOTATIONS TO ABSTRACT CONSTRAINTS

The process of abstract constraint creation is illustrated in Figure 2. The different classes of annotation are used to progressively develop scene descriptions. Setting annotations are used to segment the input text into a number of scenes. Avatar and object annotations are used to compile a list of entities that occur in each scene. Relation and transition annotations are then used to generate a list of abstract constraints that summarise the layout of the scene. Abstract constraints are then converted to mathematical constraint expressions. The solutions to the latter provide the well defined trajectories of objects in the scene.

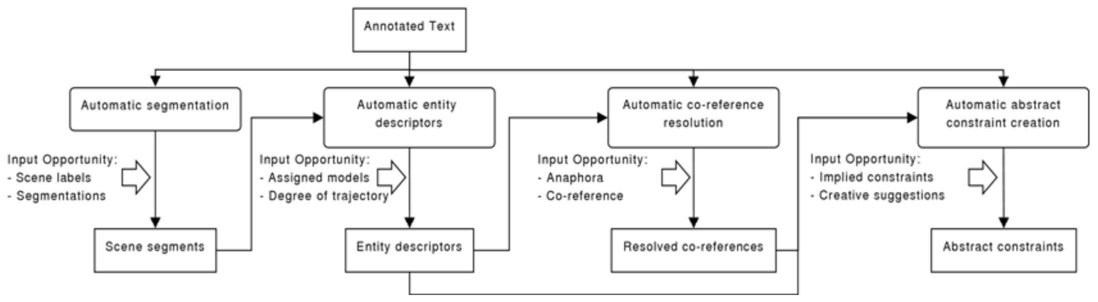


Figure 2. Illustration of the correction points for the abstract constraint creation process.

```
CONSTRAINT 1:        CONSTRAINT 2:       CONSTRAINT 3:       CONSTRAINT 4:
 Subject: MAN         Subject: MAN        Subject: MAN        Subject: MAN
 Relation: OUTSIDE    Relation: INSIDE    Relation: NEAR      Relation: NEAR
 Object: ROOM         Object: ROOM        Object: TABLE       Object: CHAIR
 Start-time: 0        Start-time: 5       Start-time: 9       Start-time: 14
 End-time: 5          End-time: 30        End-time: 14        End-time: 30
```

Figure 3. Example set of abstract constraints

An *entity descriptor* is created for every unique avatar and object annotation that occurs in a scene. Entity descriptors assign a geometric model (see Section 5), sourced from a library of pre-built models, to each avatar or abject entity. The descriptor also associates an initially unconstrained trajectory to the entity, expressed in terms of a number of variables that represent the motion of the model through 3D space. By default objects are static and are fixed at an as yet undetermined position, while avatars are assigned trajectories that permit motion.

A pointer must be created between words in the text referring to a particular entity and the descriptor for that entity. All instances of such co-reference must be resolved before relation and transition annotations can be converted to constraints. Instances of personal pronominal anaphora (such as "he") are resolved by maintaining state containing the last explicitly mentioned male and female avatars and matching the gender of the anaphora with the corresponding element in the state vector. Other co-references are resolved by matching the corresponding word with characteristics of the avatars/objects that have been previously annotated.

Transition and relation annotations are translated into abstract constraints, examples of which are presented in Figure 3. Each abstract constraint is phrased in terms of the *type* field of the annotation from which it is derived, as well as the descriptors of the involved entities. For transitions the *type* may be either ARRIVAL or DEPARTURE, while relations may be of type NEAR, INSIDE, NO_COLLIDE, BEHIND, IN_FRONT_OF, TO_LEFT_OF and TO_RIGHT_OF. *Explicit* constraints are derived directly from transition and relation annotations, while *implicit* constraints are created using an automated heuristic approach to enforce world constraints.

If the first transition constraint for an entity is an ARRIVE at some time greater than zero, then a DEPART transition constraint is inserted from zero to the start-time of the first constraint, to ensure that the entity is initially outside the scene. Any entities that are not constrained by a transition constraint are automatically assigned an ARRIVE constraint for the duration of the scene to ensure that they appear in the scene. Implicit NO_COLLIDE constraints are added for every pair of entities in the scene, lasting for the duration of the scene to ensure that entities do not interpenetrate at any point. The one exception is the case where one entity is INSIDE another entity.

A *time interval* is associated with the constraint that specifies the period over which the constraint should hold with respect to the duration of the animation. Timing information is derived using an audio narration of the text acquired from a speech synthesiser. Each language unit in the book takes a finite interval of time to be read, from which a time-line may derived for the presentation of the book. Since each scene is constructed of a sequence of tokens, the total time taken for the sounding of the corresponding audio provides a value for the scene duration, as well as a time-value for each token. Starting times for abstract constraints are derived from transition and relation annotations corresponding to the time-value of the annotated token. Ending times are based on the duration of the scene by default, but for transition constraints they are set to the starting time of a subsequent transition constraint applied to the same entity. Relation constraints may also be terminated by subsequent DEPARTURE transition constraints.

Once the abstract constraint list has been created, it may be modified manually to insert constraints regarding the scene that are not explicitly stated in the text. Opportunities exist for further creative input at several points in the abstract constraint creation process, as shown in Figure 2. The human readable format used for expressing abstract constraints supports direct modification.

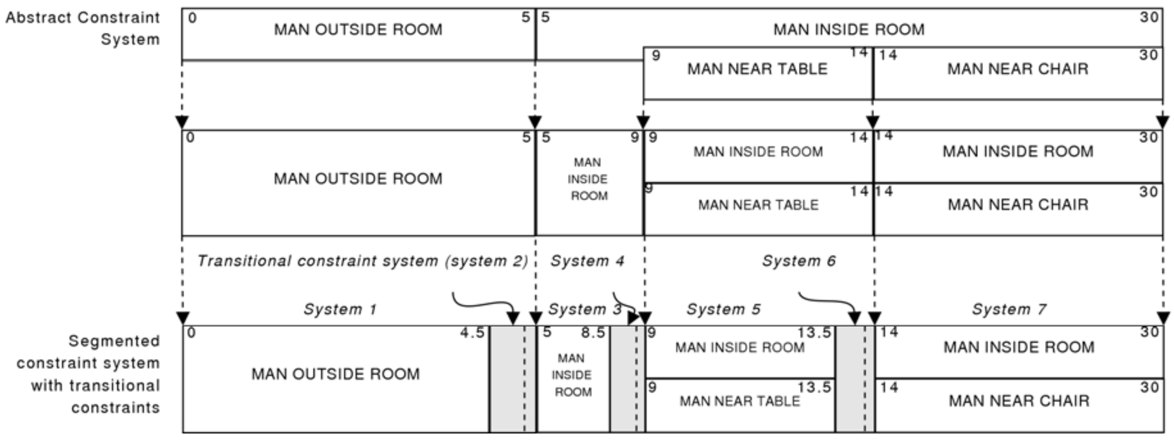


Figure 4. Illustration of constraint system segmentation

4. QUANTIFIED CONSTRAINT SYSTEMS

A set of abstract constraints such as those presented in Figure 3 is converted into a system of expressions that constrain the trajectories of the entities involved in the constraints (Glass et al., 2007). For example, MAN NEAR TABLE over interval [9,14] would be specified as a relation between the location of the two objects' bounding spheres (of radius r_{MAN} and r_{TABLE} respectively) over a specific time interval, expressed in terms of the Euclidean distance, as follows:

$$||R^{MAN}(t) - R^{TABLE}(t)||^2 < (r_{MAN} + r_{TABLE})^2 \, \forall t \in [9, 14]$$

where $R^{MAN}(t)$ and $R^{TABLE}(t)$ are the trajectories of MAN and TABLE as functions of time respectively.

Trajectories are expressed as curves of degree *n*. The higher the degree the more difficult the constraint solving process becomes. To counter this, the trajectory of each model is segmented into chains of lower degree curves (Christie et al., 2002). At present we find that a satisfactory trade-off between performance and quality is achieved with the use of chains of first degree curves.

Figure 4 illustrates graphically the abstract constraint system of Figure 3 and indicates how the constraints are segmented into a chain of 7 constraint systems. Initially a unique system is created for each contiguous time interval, by dividing the duration of the scene into intervals. For each interval, a single set of constraints is applicable over the full duration of that interval. A *transitional* system is inserted between adjacent intervals allowing a period during which both sets of constraints must be satisfied so that model locations blend smoothly from one interval to the next. To facilitate efficient constraint solving, the ending locations for each interval are used as starting locations for solving the following constraint system.

Constraint solving establishes the trajectory for each entity, which may then be traced in a 3D space, defining the motion of the model through a scene. Constraints are phrased over a universally quantified time interval, and this in conjunction with an interval-based constraint optimization algorithm (based on the universally quantified constraint solver developed by Benhamou et al. (2004)) guarantees that a solution is valid for every point during the time interval, even when the constraint systems are non-linear. A benefit of the optimization-based approach is that it provides approximate solutions even for constraints resulting from inconsistent annotations, and also produces useful intermediate solutions if limited processing time is available.

(a) Constraint system illustrated as weighted graph

(b) Three incremental constraint systems

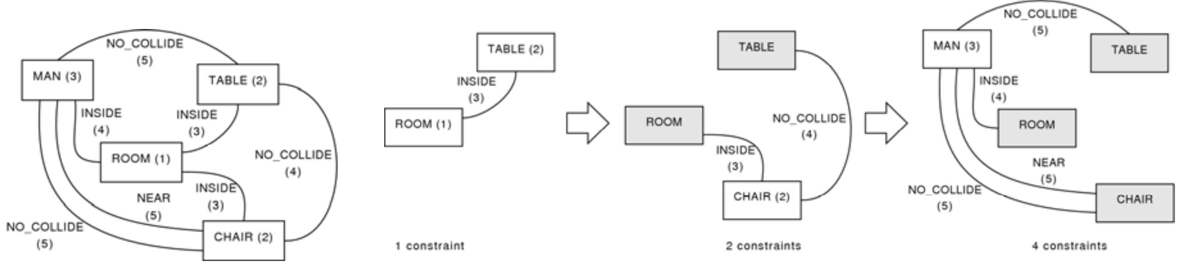


Figure 5. Illustration of incremental constraint solving

Each constraint system resulting from the segmentation process is solved in an incremental fashion. All constraints involving static objects are solved first, followed by those involving dynamic objects. This is achieved by representing the scene as a weighted graph where nodes represent entities, and edges represent constraints. Edges are given increasing weights according to whether the adjacent entities are scenes, objects or avatars. The order in which constraints are solved is determined by the weights of the corresponding edges. Figure 5 illustrates this process, where in each step the highlighted entities represent trajectories for which solutions have already been found.

We find that trajectory chaining in conjunction with incremental system solving results in constraint systems that can be rapidly solved using the quantified interval-based constraint optimization algorithm. The results of this step are well defined trajectories for all entities.

5. AUTOMATIC ANIMATION ASSEMBLY

Once trajectories have been quantified for a set of scenes, we automatically create a graphical representation of the annotated text. The 3D environment is created and populated and the final animation is sequenced.

Humanoid models are chosen for avatars, while object models are located automatically by matching the annotation with keywords associated with each model in the library. The annotated token associated with an object is used as search term when querying the model library. If a model's keywords match the token then the model is chosen to represent the object. If no matching keyword exists, then all the synonyms of the term returned using WordNet (Fellbaum, 1998) are tried. If no match exists in this case then the immediate hypernym of the term is used as a search term. The process of generalisation continues until the term can no longer be generalised, and a default placeholder object is selected (a cube).

We make use of the setting annotation to automatically create background geometry using procedural methods. In particular, three types of setting are defined, namely *terrain*, *room,* and *city*. Using WordNet (Fellbaum, 1998), a setting annotation may be associated with one of these categories. In particular if a setting annotation is a hyponym of the term "geological formation" or "geological area" it is classified as a *terrain*. Any term with the hypernym "urban area" is classified as a *city*, while any term with the hypernym "room" is classified as a *room*. Terrain geometry is created automatically using a method adapted from recent work by Belhadj (2007), which allows the creation of realistic looking terrain around pre-defined points. Using this approach terrain can be specified to "support" all entities in the scene. Procedural city generation is implemented based on work by Parish and Muller (2001), but at this stage this method only produces gridiron road patterns, with textured cubes as buildings. Rooms are generated through the creation of geometry defining four walls and a floor, textured, and containing openings where entities enter or exit the scene.

Animation assembly is performed in the Blender modeling package[1] which is an open source 3D modeling and animation tool that provides a Python scripting interface as well as a video sequencing editor. The scripting interface is used to convert trajectories and object descriptors produced by the previous stages into 3D environments.

Blender allows for the creation of multiple 3D environments, conveniently called *scenes*. As such, every scene identified in the text is created as a separate 3D environment in Blender, from which segments may be rendered and placed into the video sequencer along with subtitles and audio. Subtitles, audio narrations and foleys are also generated automatically (Glass et al., 2007).

The model library is an archive of existing Blender files containing object geometry, materials, armatures and motion capture data which can be linked into new scenes. Models in the library may be modified by the scripting environment to create customised characters for each avatar, changing clothing and hair colouring for example.

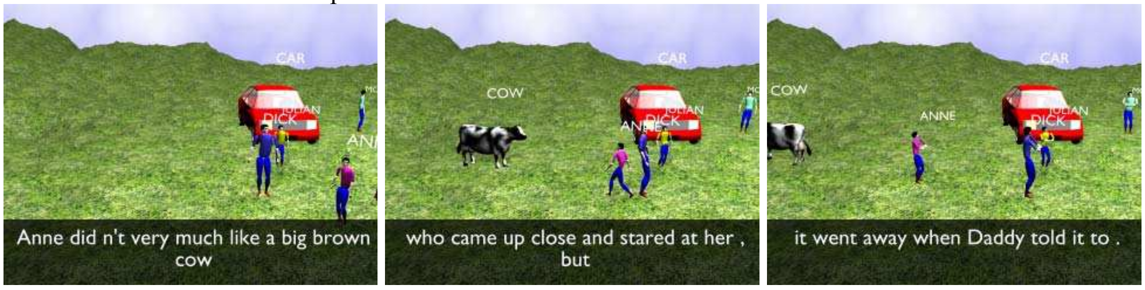
Positioning and motion within the 3D environment is controlled using Blender's interpolation (IPO) curve structure, which defines a model's translation in a scene in terms of an independent curve for each dimension. Each sampled location point from the entity's trajectory defines a point on the corresponding IPO curve. The orientation of each model is modified concurrently to always face the direction of motion. Depending on the velocity of the model at each point, the appropriate pose (for example stand, walk or run) for the models is selected automatically.

6. RESULTS

We present extracts from a fiction book that have been converted to animation sequences using the process described in this paper. Figure 6 presents snapshots based on the annotated text example of Figure 1. Figure 7 presents another example from the same book.

The nature of the manual intervention is indicated in each figure. Examples of the changes that have been applied in each stage of the process include: re-annotating the cow as an avatar rather than a static object in Figure 6; manual resolving of the pronoun "it" to the cow object in Figure 6 and distinguishing between the two male avatars in Figure 7; and removing extraneous models in the scenes created in Blender. In comparison to the amount of effort that would be otherwise involved in creating these scenes manually, these changes are accomplished quickly and easily.

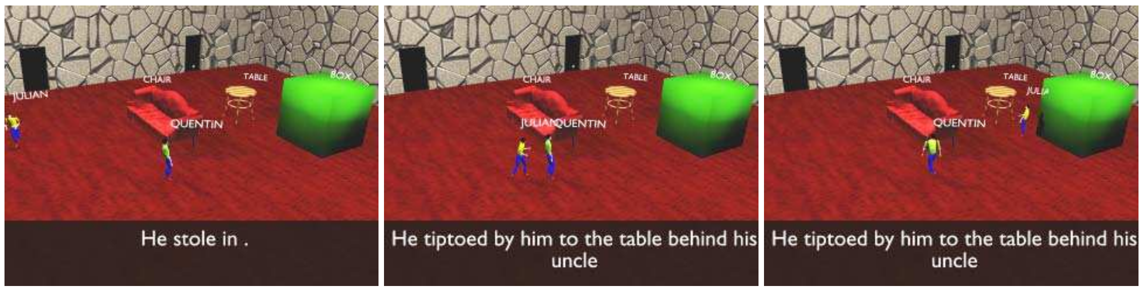
The presented examples are ideal in that they contain explicit instances of the particular annotation categories. In extracts containing less explicit descriptions, the process still generates a 3D animation, however the correlation between the descriptions and the animation may be more difficult to identify. In these cases additional manual input is advisable.


CAR
JULIAN
DICK
ANNE
Anne did n't very much like a big brown cow
COW
CAR
JULIAN
DICK
ANNE
who came up close and stared at her , but
COW
ANNE
CAR
JULIAN
DICK
it went away when Daddy told it to .

Manual intervention: 1 of 12 model descriptors modified; 2 of 12 co-references corrected; 0 of 68 abstract constraints added; 3 models deleted from 3D scene.

Figure 6. Cow scene from *The Famous Five: Five on a Treasure Island* by Enid Blyton

[1]Available at http://www.blender.org/


JULIAN
CHAIR
TABLE
BOX
QUENTIN
He stole in .
JULIANQUENTIN
He tiptoed by him to the table behind his uncle
JULIA
He tiptoed by him to the table behind his uncle

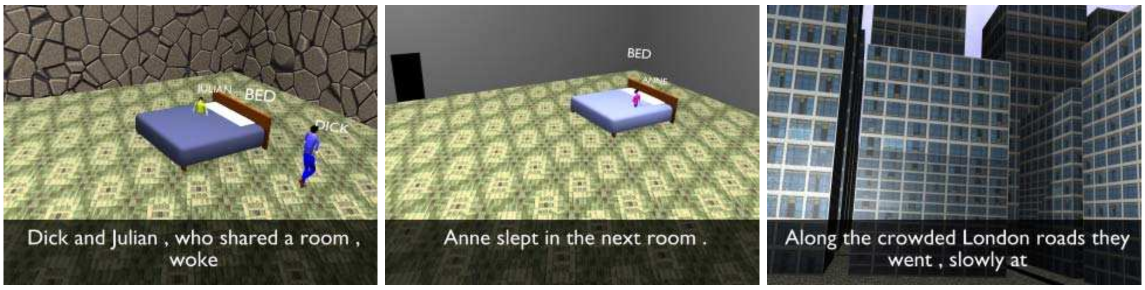
He stole **<transition type='ARRIVAL' subject='He'>in</transition>**. His **<avatar>uncle</avatar>** still snored. He tiptoed by him **<relation type='NEAR' subject='he' object='table'>to</relation>** the **<object>table</object> <relation type='BEHIND' subject='table' object='chair'>behind</relation>** his uncle's **<object>chair</object>**.

Manual intervention: 1 manual specification of setting (namely, "study"); 0 of 8 model descriptors modified; 8 of 40 co-references resolved manually; 0 of 45 abstract constraints modified; 1 model deleted from 3D scene.

Figure 7. Study scene from *The Famous Five: Five on a Treasure Island* by Enid Blyton

Figure 7 presents an example in which a *room* setting is created, and which requires the correct layout of objects in the scene. Notice that the table is behind the chair as specified, and how the model representing Julian moves to the correct locations according to the annotations. The motion of avatars is continuous but their speed is erratic due to the intervals allocated which are completely determined by the time taken to read the corresponding fragment of text.

An additional extract is presented in Figure 8 demonstrating the automatic creation of an animation containing multiple scenes, including two distinct *rooms* and a *city*. Figure 8 is an example in which human creativity is applied in the form of an additional abstract constraint specifying ANNE INSIDE BED, since this fact is not explicitly stated in the text. This demonstrates that world knowledge may be easily applied to an automatically created scene through manual intervention, removing the requirement for a complex knowledge-base.


JULIAN
BED
DICK
Dick and Julian , who shared a room , woke
BED
ANNE
Anne slept in the next room .
Along the crowded London roads they went , slowly at

<avatar>Dick</avatar> and <avatar>Julian</avatar>, who shared **a <setting>room</setting>**, woke up at about the same moment, and stared out of the nearby window. … **<avatar>Anne</avatar>** slept in the next **<setting>room</setting>**. **<avatar>Julian</avatar>** ran **<transition type='ARRIVAL' subject='Julian'>in </transition>** and shook her.…Along the crowded **<setting>London</setting>** roads they went, slowly at first…

Figure 8. Travel sequence from *The Famous Five: Five on a Treasure Island* by Enid Blyton

7. CONCLUSION

We present examples of 3D animated scenes constructed from annotated fiction text. The process readily allows for creative input from human directors to add artistic vision and external world knowledge. Abstract constraints are derived from suitably annotated fiction text and these are used to restrict the form of object trajectories within the scene. We outline a process for determining these trajectories using a quantified interval-based constraint optimization algorithm that produces solutions valid over continuous time intervals while bounding the amount of processing required. Annotations and trajectories are used in combination with a detailed model library for the instantiation of 3D environments. To our knowledge the research presented here is the first to describe a system that converts extensive extracts from popular fiction into corresponding animations.

Future work includes investigation into additional annotation categories, and further improvements in the generation of procedural models.

REFERENCES

Badler, N.I. et al., 2000. Parameterized action representation for virtual human agents. *Embodied conversational agents.* MIT Press, Cambridge, USA, pp. 256–284.

Belhadj, F., 2007. Terrain modeling: a constrained fractal model. In *AFRIGRAPH '07: Proceedings of the 5th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa.* New York, USA, pp. 197–204.

Benhamou, F. et al., 2004. Interval constraint solving for camera control and motion planning. *ACM Transactions on Computational Logic (TOCL)*. Vol 5, No. 4, pp. 732–767.

Christie, M. et al., 2002. Modeling camera control with constrained hypertubes. In *CP '02: Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*. London, UK, pp. 618–632.

Clay, S.R. and Wilhelms, J., 1996. Put: Language-based interactive manipulation of objects. *IEEE Computer Graphics and Applications*. Vol 16, No. 2, pp. 31–39.

Coyne, B. and Sproat, R., 2001. Wordseye: an automatic text-to-scene conversion system. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. New York, USA, pp. 487–496.

Fellbaum, C. editor, 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, USA.

Glass, K. and Bangay, S., 2006. Hierarchical rule generalisation for speaker identification in fiction books. In *Proceedings of SAICSIT '06*. Republic of South Africa, pp. 31–40.

Glass, K. and Bangay, S., 2005. Evaluating parts-of-speech taggers for use in a text-to-scene conversion system. *Proceedings of SAICSIT '05*. Republic of South Africa, pp. 20–28.

Glass, K. et al., 2007. Mechanisms for multimodality: taking fiction to another dimension. In *AFRIGRAPH '07: Proceedings of the 5th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*. New York, USA, pp. 135–144.

Johansson, R. et al., 2005. Automatic text-to-scene conversion in the traffic accident domain. In *IJCAI-05: Proceedings of the 19th International Joint Conference on Artificial Intelligence*. Edinburgh, Scotland, pp. 1073–1078,

Joshi, D. et al., 2004. The story picturing engine: finding elite images to illustrate a story using mutual reinforcement. In *Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval.* New York, USA, pp. 1073–1078.

Lu, R. and Zhang, S., 2002. *Automatic Generation of Computer Animation: using AI for movie animation*, volume 2160 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, Germany.

Ma, M., 2006. *Automatic conversion of natural language to 3D animation*. PhD thesis, University of Ulster, Derry, Ireland.

Parish, Y.I.H. and Muller, P., 2001. Procedural modeling of cities. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques.* New York, USA, pp 301–308.

Zeng, X. et al., 2003. Shape of the story: Story visualization techniques. *IV'03: Proceedings of the 7th International Conference on Information Visualization*. pp. 144–149.

Zeng, X. et al., 2005. Gough. From visual semantic parameterization to graphic visualization. In *IV'05: Proceedings of the 9th International Conference on Information Visualisation*. Washington D.C., USA, pp 488–493.