Improving Communication Using 3D Animation

Laurent Ruhlmann, Benoit Ozell, Michel Gagnon, Steve Bourgoin, Eric Charton

Ecole Polytechnique de Montréal, 2900 boulevard Edouard-Montpetit, Montréal, QC H3T 1J4, Canada
{laurent.ruhlmann, benoit.ozell, michel.gagnon, eric.charton }@polymtl.ca, sbourgoin@unimasoft.com

Abstract. This paper presents a high level view of a project which aims at improving the communication between people who do not share the same language by using a 3D animation upper layer to lift the inherent limitations of written text. We will produce animation using the Collada file format, a standard based on the XML format. This project (GITAN) has started in January 2010, and we expect to have the first results by the end of 2010. A limited set of sentences will validate the global process and help us refine the tools developed in the pipeline.

Keywords: Animation, Collada, Natural Language Processing, Ontology

1. Introduction

Since Aristotle, human language has been considered a partial way of representing the human thoughts. "The soul never thinks without a mental image"[1]. With the development of computer graphics, a new family of tool has emerged to improve communication. In this paper we present the *Gitan Project.* This project aims at creating a new language based on animation sequences in order to swap between text and graphics. More specifically, the project will develop a grammar linking text and animation, thus allowing the conversion between them. Such a grammar will allow us to lift the inherent ambiguity of natural languages. Using the powerful multimedia capacity of animation, it will be able to move from an abstract representation to a more concrete graphical view. There are many challenges that this project will face; one of the most important one is the ability to maintain the number of the existing animation segments at a reasonable level. Another important challenge is the fact that we want to combine algorithmically various animations into a plausible solution. When combining a walk cycle and a translation, in order to represent a walking person, we must ensure that the duration of the animation and the speed of the walk represents a good approximation.

2. Related work

Animated scenes have already been used as a good medium to help the communication between adults and children [2], to illustrate domain specific activities like car accident simulations [3], or even for storytelling [4]. Another important project was aimed at the object reuse and the adaptation of 3D data for automatic scene creation [5]. The major difference between those projects and this one comes from the application domain and the linguistics approach we are taking. We want to cover a broader set of words, not limiting ourselves to a specific domain like car accidents. We want also limit the number of original animations to a minimum. In the first phase, we will not handle emotions or high level of languages like poetry.

One of the hypotheses of this project is that we will be able to reconcile the text and the animated image which is distinct types of representations [6]. We believe that animations will facilitate the understanding of a text, in different languages. Another hypothesis is that the principle of compositionality [7] is also applicable to animations. Being able to compose various animations in order to produce a plausible sequence is key to the project; we have already validated some simple cases. A third hypothesis is that the computer graphics field is mature and rich enough to be used as a good communication medium. We are not planning to develop videogame-like animations, but the realism achieved by some current games gives us some confidence about the validity of this hypothesis, even if the goal of the project is communication and not realism; In fact too much realism could limit effective communications between non-native speakers, since each culture has its own visual standard for representing emotions or complex actions. A final major hypothesis is that through proper usage of semantic tools (like ontologies), applied to the computer graphics domain, we will be able to minimize the amount of objects and animation to be modeled , in order to produce meaningful animations.

3. The Animation standards

The term animation has been used throughout the ages to define ‘rapid display of a sequence of images of 2-D or 3-D artwork, in order to create an illusion of movement’ [8]. It is interesting to note that in 3200 BC, a bowl created with a sequence of 5 images depicting a jumping goat was found in Iran. Since it is easier to use a computer, we will limit our definition to 3D artwork generated and displayed by a computer. In this domain, we will privilege key-frame based animation, since this is the most common used technique, and realism (which is for instance achieved by physically based animation) is not our primary goal. In order to be open and readable by many Digital Content Creation (DCC) software, we will use the Collada (COLLaborative Design Activity) format, which is emerging as a new exchange standard [9]. DCC software (3DS Max, Maya, Softimage, 3dvia,) are already supporting it. Many viewers supporting it are available. It is an XML schema supporting animations, physics and many other features. It allows extensions which will be used to define specific information needed by the project. There are a lot of scripting languages (Maya's MEL, Softimage ICE, Virtools’s SDK, 3ds Max's Maxscript, Unreal’s Development Kit ...), but they fail to provide a high level

language which will allow the complete description of an animation for our project. They all have very good scripting capabilities, and are focused on areas which intersect only partly with our goals. This is why we decided to develop our own formalism and language.

4. Intermediate format between text and animation.

Going from text to animation is a complex process which cannot be achieved in one step. We developed an incremental approach, using an intermediate representation based on the XML schema concept. It is an XML schema [10], since this formalism provides a clear and easy way to specify the various elements of a grammar. The file based on this format is split in 2 main parts, the *Data* and the *Process* parts. The *Data* part contains a detailed description of all the elements needed in the animation, and the *Process* part describes how the animation is going to be played. It can be viewed as a textual animation storyboard [11**].** A detailed description of the parts is given in section 6.1 and specific examples in section 6.8.For a complete description of the project's architecture, see figure 1.

5. The need of ontologies

There are many definitions of the term ontology depending on the domain to which it applies. For our purpose we will use the definition used in the information science: "An ontology is a specification of a conceptualization." [12]. We will use the ontologies tools (like OWL, [13]), to abstract meaning from the geometries used in the animations. This way we will be able to minimize the amount of 3D objects needed for building the animations. The most interesting project to create and define semantic based models for digital objects is the AIM@SHAPE project [14]. Due to its wide spectrum , it has raised a lot of interest among the semantic web community [15]. This is an avenue where the project may want to invest in the near future, once the animation ontology is mature enough. We are planning to use an ontology for static objects in the scene. This way the word 'Boeing', as well as 'Cessna' and airplane, will be mapped to the same 3D representation of a plane.

6. Architecture of proposed system

This section describes in details how we are going to produce 3D animations which will carry all the textual information. We give an overview of the architecture and then detail the 2 modules which will produce the final animation. A description of the various data needed for the process is also given.

6.1. Architecture components

Our system is based on 3 main components. The **Language Engine** produces a semantic form of the text. The **Text To Scene Engine** converts then this annotated

text in an animatable format: the animatable scene. This is the input to the **Graphics Engine** which will produce a Collada file, consumable by any compliant viewer. An important part of the project is **the GITAN Repository**, containing partial Collada files which will be part of the final Collada scene.

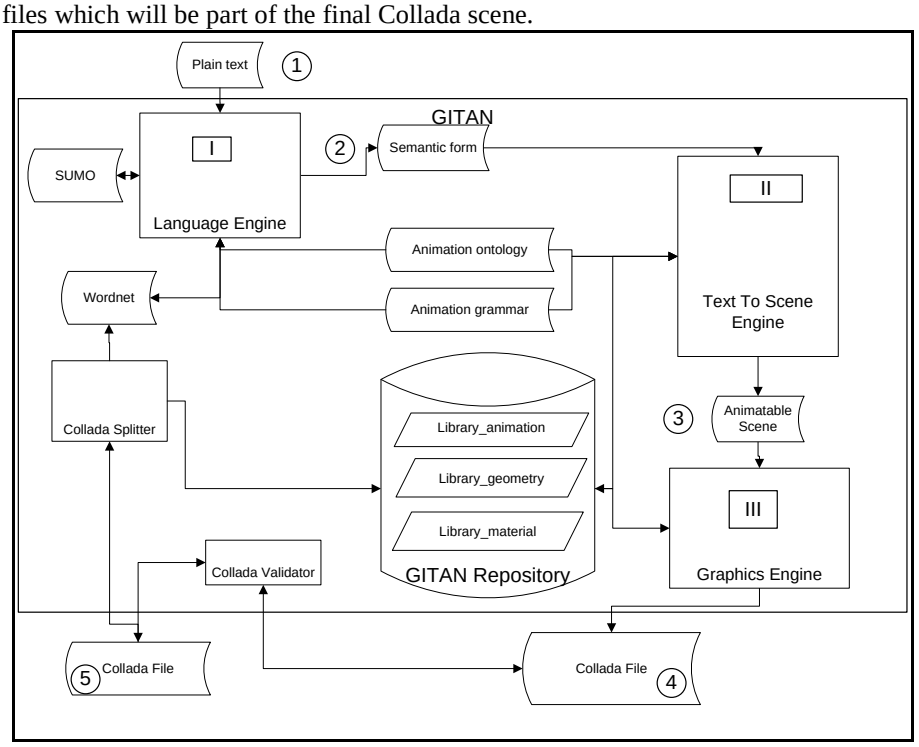

Figure 1 Gitan's architecture

6.2. The Gitan Repository

The GITAN repository is the storage containing atomic elements of a Collada file encapsulated with GITAN specific meta-data. The geometries describe the mesh structure of the object, plus GITAN tags (specifying whether this geometry static or animatable is, for instance). The animations follow the Collada animation specifications [9], and contain also GITAN meta-data (like the origin of the Collada segment, its expected input for an activity). It contains all the elements to build a Collada file: cameras, lights,.. It must be noted that we aim to keep this repository as small as possible, by parametrizing as much as possible all the segments. We must be able to morph a cat into a tiger, provided the necessary information. The next section describes how the semantic form of the text is generated.

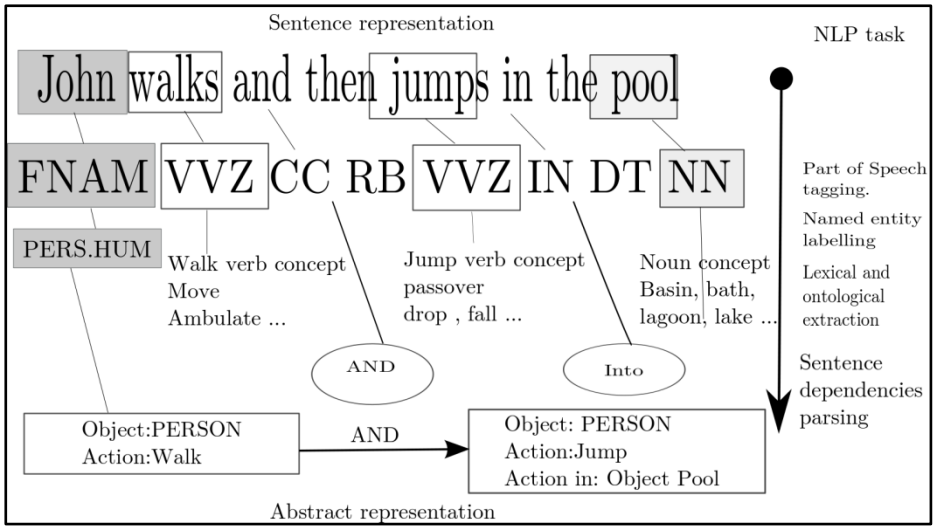

Figure 2 : The Language Engine

6.3. The Language Engine

The language engine transform a sentence into a semantic representation. Generic Natural Language Processing methods are used to achieve this transformation. First a labelling process involving a Part of Speech Tagger is applied. Part of speech helps to identify nature of textual information like verbs, adjectives and nouns in the perspective of their animation. Then a Named Entity (NE) labelling step is performed. This step allow identification of specific entities like Persons, Location, Products. Then each NE, verb and nouns are linked with an ontological description. This allows to associate its exact sense with a noun or an entity (i.e. A boat or a building defined in the ontology with their instance word inside the sentence), a specific movement with a verb (i.e. the sense of *course* for the verb *Jump* in a text context *of movement*) or to apply an attribute to a an object (i.e. using adjectives and link their description to their the concerned ontological instance). The labelling process and algorithms to instantiate relation between a word and its ontological representation have been experimented [16]. Final step involve hierarchical representation of terms dependence and identification of semantics sense inside the sentence (i.e. *the cat eats the mouse* or *the mouse is eaten by the cat* have the same sense but use different relation schema). This work have been investigated in [17].

6.4. The Text To Scene Engine

This module converts the semantic representation into an animatable description of the sentence. The major modules are the ones replacing absent entities and converting events into constraints and activities. The GITAN repository is searched in order to

find an object or an animation fitting the ontological instance of a word. If present, a tag is created in the file. If absent, the GITAN ontology, using the OWL format [13], will provide a proper link to a potential candidate: if we want to instantiate a tiger in the animation for instance, and if the repository contains a cat, this module will morph the cat into a tiger by scaling it and adding a proper texture to it. The GITAN ontology provides the various parameters to be modified (i.e. size and texture). The same principles apply to the ontological instances of events. If the event is absent from the repository, the ontology will infer a proper parameterized substitute if possible. A walk cycle can potentially be used as a substitute for a jump cycle. Using the GITAN grammar, the TTS engine will link and build the tags in the animate text file. Using constraint calculus [18] , we aim to provide a valid transition for action verbs, and the global positioning of objects in the scene.

6.5. The Graphics Engine.

Its goal is to assemble the pre-existing objects (defined as Collada segments in the GITAN repository) into an initial scene and apply the animations to them, according to the definitions from the Animatable Scene input file The most important parts of the engine are the constraint converter and the scene builder. The constraint converter will translate into a Collada format all the constraints and activities found in the Animated Text file. The scene builder will create the Collada scene, using objects stored in the GITAN repository. It will also assign the animations to the objects. The input data used by the graphics engine is made of 2 distinct types: The animatable scene and the Collada segments. The Collada segments are stored in the GITAN repository, and are programmatically included in the final Collada file.

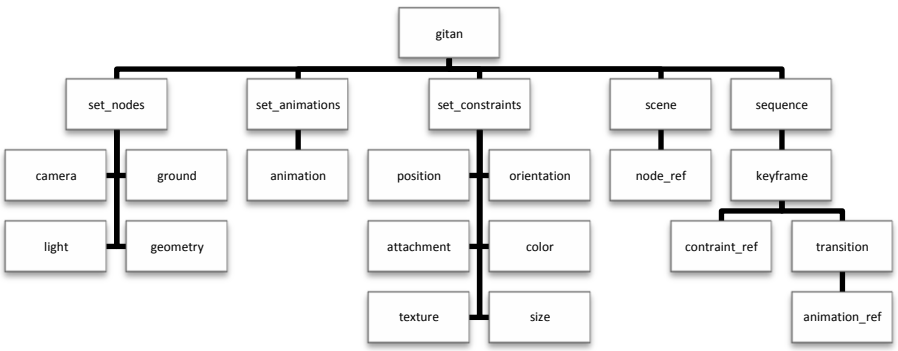

Figure 3: the animated scene file

*The animated scene file***:** this input contains a high level scene description, based on a XML schema. This is a constraint based key-frame set, defining the various objects needed in the scene, and a set of key-frames, which will trigger animations and constraints. It is made of 2 sub parts: The *data* and the *proces***s** parts. Data describes all the needed objects, animations, for creating the scene. The process part

specifies the timing of the animations and the constraints existing between static objects. .

The data part contains sets of nodes, animation and constraints. The *nodes* define static objects from the GITAN repository. The *animations* can be a Collada animation; it defines then a predetermined GITAN animation, stored in the repository. A good example is a walk cycle, since such an animation is too complex to be described in a high level animation file. The *animation* can also be defined as a displacement. In this case it will be specified in the animated scene file, by its type (linear, ballistic ...), its speed. This part of the file also contains complete descriptions of the object's *constraints*. We identified 6 major constraint types affecting the position, the color, the size, the orientation, the texture of an object. There is also an attachment constraint, defining a link between 2 objects. An *animation grammar* will be produced which will convert the annotated verbs into animatable constraints. See figure 4 for details.

The process part defines how the scene is to be built and what constitutes the animation. We are using keyframed animation [19], since it's a common standard among DCC and Collada supports it. It contains 2 main components: the *Scene* and the *Sequence*. The Scene is a static description of the scene to be animated. The Sequence is made of *ke***y***frame***s** and *transitions*. A *keyframe* defines the object's static constraints (positional, color ...) at a certain time. A *transition* defines how the objects change (its position, its shape...) from one keyframe to another. A transition must be made of at least *1 animation*, named the major animation. It can also contain secondary animation. Figure 4 illustrates the link between the various part of the file and the GITAN repository through a specific example ('the man jumps in the pool'). The <*Man*> node is linked to the 'Man' Collada geometry, and is positioned in the Keyframe 1. The <*Pool*> node has the same role. The <*walk-activity*> links the 'Walk cycle' Collada Anim1 to the Transition 1 where it will be executed, like the Translate Displacement, translating the <*Man*>. This creates a translation of the object Man, using a Collada walk cycle. At Keyframe2, the final Man position is set, through the <*Man Position Constraint2*>. The < *walk-activity*> is also linked to the <man> object, since this Collada animation must be applied to a valid Collada object, the 'Man Collada Geom1' in this case.

6.6. Development

Since the goal of the project is long term, and there are many areas of research which could influence its development, we will use a staged approach, based on an initial prototype working on a limited set of words. This will help validate one our original hypothesis (regarding the unification of text and image).

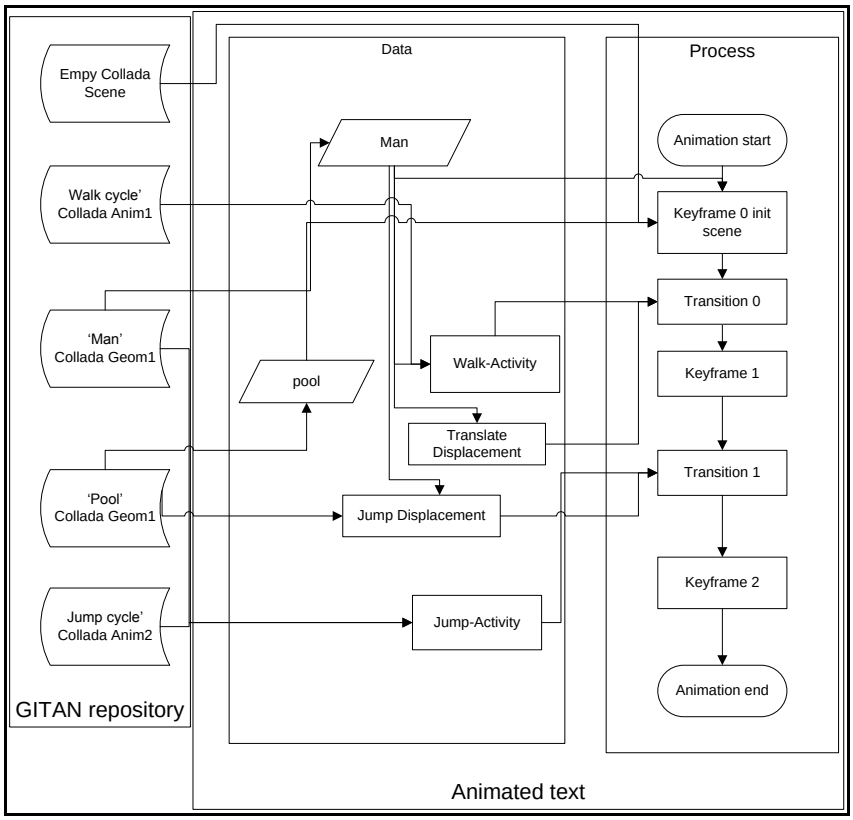

Figure 4: the animatable scene file, the GITAN repository

6.7. Initial prototype.

The first prototype will revolve around a specific use case: the teacher types a simple sentence. An animation is created. The teacher validates the animation, associated to a bag of word. The student sees the animation, and using the same set of words assembles them, in order to produce the same animation as the one he saw. This prototype will validate the Graphics engine, and the GITAN repository access. For this prototype all words and all the animations representing the verbs will be available in the GITAN repository;

6.8. Preliminary results

We have validated the principles described by manually annotating all the elements needed by the Graphics engine. Here is

The original sentence: "John walks and then jumps in the pool"

The animated text contains the 2 parts *data* and *process :*

data

object 1, 'john' , URI = c:\data\models\man.gitan

object2 , 'pool', URI = c:\data\models\pool.gitan

animation-activity1 : 'walk-cycle' , URI=c:\data\animation\biped-walk.gtn.

animation-activity2 : 'jump-cycle', URI = c:\data\anmation\jump-cycle.gtn

animation--displacement1: translation, object1 //moves object1 linearly

animation-displacement2: jump, object1 // moves object ballistically

constraint1: 'position1' , object1. // place object1 at position1

constraint2: 'position2', object1. . // place object1 at position2.

constraint3: 'position3', object1. . // place object1 at position3

constraint4: 'position1', object2. . // place object2 at position1

process

keyframe1: contraint1, constraint4.

transition1 (10 seconds**)**

animation-displacement1 (major , 4km/h)

animation-activity1 (minor, 1 cycle/sec)

keyframe2: constraint2

transition2 (20 seconds)

animation-displacement2 (major, 6km/h)

animation-activity2(minor, 1 cycle/sec)

keyframe3: constraint4

The *displacements* (jump, translation. etc...) are all predetermined, having specific parameters. A jump will have the distance and height attribute for instance. The *constraints* are of predetermined types: positional, colour, orientation. All those functionalities will be refined during the progress of the project.

7. Conclusions

With this project we want to liberate the communication between people of the textual or oral language limitations, by using 3D animations. This paper described the various modules which are going to be developed, for the graphical part, which will produce 3D animations. There are open questions that will be solved during the development of this project: how to maintain automatically the integrity of the Collada segments in the repository? Is the 'simple animation' (i.e. not realistic like a video game) paradigm conveying properly the communication intention between 2 persons? Is the proposed keyframe-transition model based on constraint calculus general enough ? the project starts and promises to be very exciting due to all the challenges we face. We want to thank Prompt [20] and the UnimaSoft Company [21], which are funding this project.

8. References

1. Aristotle, De Anima, 431a, 15-20.
2. Kaoru S, Mizue N: Animated Storytelling System via Text in: ACE 06, June 14-16, 2006, Hollywood, California, USA.
3. Dupuy S., Egges A, Legendre V, Nugues P (2001) Generating a 3D Simulation of a CarAccident From a Written Description in Natural Language: The CarSim System. *In Proc. of the Workshop on Temporal and Spatial Information Processing, 1-8, ACL 2001 Conference, Toulouse, 7 July.*
4. Ma, M: Automatic Conversion of Natural Language to 3D Animation, PhD thesis, University of Ulster.
5. Bilasco I, Villanova-Oliver M, Gensel J, Martin H : Semantic-based rules for 3D scene adaptation in *Web3D '07: Proceedings of the twelfth international conference on 3D web technology , April 2007*
6. Denis, M. Imagery and thinking. In: C. Comoldi & M.A. McDaniel (eds.), Imagery and Cognition. New York: Springer-Verlag, 1991.
7. Frege, F. L. G. Über Sinn und Bedeutung. Zeitschrift für Philosophie und philosophische Kritik, 100 (pp. 22-50). Trad. fr. Claude Imbert, « Sens et dénotation », *in Écrits logiques et philosophiques (pp. 102-126), Paris: Seuil, 1971.*
8. Wikipedia [http://en.wikipedia.org/wiki/Animation]
9. COLLADA [www.collada.org]
10. W3C schema definition page site [http://www.w3.org/standards/xml/schema]
11. Wikipedia [http://en.wikipedia.org/wiki/Storyboard]
12. Tom Gruber [http://tomgruber.org/writing/ontology-definition-2007.htm]
13. OWL [http://www.w3.org/TR/owl2-overview/]
14. AIM@SHAPE [http://www.aimatshape.net/]
15. Linked Data [http://linkeddata.org/]
16. Eric Charton, Michel Gagnon, Benoit Ozell (2010) Extension d'un système d'étiquetage d'entités nommées en étiqueteur sémantique In P*roceedings of TALN 2010, Montréal, Canada*
17. Amal Zouaq, Michel Gagnon and Benoit Ozell (2010): Semantic Analysis using Dependency-based Grammars and Upper-Level Ontologies, In *Proceedings of the 11th International Conference on Intelligent Text Processing and Computational Linguistics*, 2010.
18. J. Renz, B. Nebel, Qualitative Spatial Reasoning using Constraint Calculi, in: M. Aiello, I. Pratt-Hartmann, J. van Benthem, eds., *Handbook of Spatial Logics*, Springer Verlag, Berlin, 161-215, 2007
19. Foley-Van Dam, computer graphics, Principles and practice, 2nd edition, Addison-Wesley, 1990
20. PROMPT [http://www.promptinc.org/]
21. UnimaSoft ['http://www.unimasoft.com]