Arjan Egges, Wolfgang Hürst, Remco C. Veltkamp (Eds.)

Proceedings of the CASA Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS)

Amsterdam, The Netherlands, 16 June 2009

Arjan Egges
Wolfgang Hürst
Remco C. Veltkamp
(Eds.)

Proceedings of the CASA Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS)

Amsterdam, The Netherlands, 16 June 2009

ISBN 978-90-393-5102-4

Table of Contents

Preface

The 3AMIGAS workshop is about the creation and use of advanced 3D media in gaming and simulation. Simulations and games take place in virtual worlds. Such worlds can either be models of existing environments or can be fictitious. They can have a very realistic look-and-feel or they can be imaginary. To create adaptive game play and simulations, dynamic world models are required that can automatically be adapted to changing requirements. Virtual worlds are inhabited by virtual characters. These characters must behave in natural ways, based on the way they sense their environment and based on their own internal emotional state and relations to other characters. The virtual characters should be able to adapt to the situation at hand. They should react and plan their actions, and smart objects should react to interaction. In augmented worlds, natural navigation and interaction is a real challenge due to limitations in multimodal control and feedback technology and due to limitations in the understanding and true-to-life modeling. Traditional interfaces are generally not natural and demand substantial cognitive resources. All these issues are aspects of advanced 3D media.

We have broadcasted an open call for papers to attract a representative number of papers from leading researchers working on topics related to 3D media in gaming and simulation, including but not limited to navigation, smart objects, formalization and representation of shape, semantics, virtual characters, content-based 3D retrieval and classification, automatic creation of imaginary worlds, semantics-driven 3D visualization, 3D media ontologies, automatic world generation based on real data, semantics-based 3D modeling, interaction in 3D worlds. The present proceedings contain two invited keynotes, five reviewed technical papers, and one position paper

The workshop is partially supported by the project FOCUS K3D funded by the EC FP7, and the GATE (Game Research for Training and Entertainment) project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie). We like to thank all our sponsors, the members of the program committee, all authors, the local organization, and anyone else who contributed to making this workshop a success.

Arjan Egges, Wolfgang Hürst, Remco Veltkamp

(Utrecht University, The Netherlands)

June 2009

Program committee

Rafael Bidarra, Delft University of Technology, the Netherlands
Roland Geraerts, Utrecht University, the Netherlands
Catherine Houstis, CERETETH, Greece
Nadia Magnenat-Thalmann, University of Geneva, Switzerland
Stephane Marchand Maillet, University of Geneva, Switzerland
Michela Mortara, CNR-IMATI, Italy
Stijn Oomes, Delft University of Technology, the Netherlands
Pietro Pala, University of Florence, Italy
Laura Papaleo, Univ. Genoa, Italy
Ewald Quak, SINTEF, Norway
Patrick Salamin, EPFL, Switzerland
Michela Spagnuolo, CNR-IMATI, Italy
Andre Stork, Fraunhofer, Germany
Daniel Thalmann, EPFL, Switzerland

Collaborative Visual Analytics through Serious Gaming

Johan de Heer
T-Xchange - Engineering Innovation
Enschede, The Netherlands
Johan.deHeer@TXchange.nl

Abstract

We focus on the applicability of serious gaming facilitating complex decision making processes. In particular, we introduce the T-XChange and touch upon the Effect Based Solution (EBS) process, in which gaming takes a significant part. EBS will be presented as a technology-mediated facilitation and capturing method for multi-stakeholder assessment, planning and/or decision-making processes.

The Effect Based Solution method and apparatus provides a series of steps for eliciting - from various multi-stakeholder perspectives - the important aspects associated with a particular problem- and solution space about which a concept solution is to be devised. The stakeholders interact with (mixed reality) serious games which are designed to pictorially represent important wicked problem- and innovative solution aspects of a topic being studied. The EBS process can be viewed as a multi-stakeholder multi-criteria decision-support system.
The serious games allow for 'collaborative visual analytics' and are then used to interactively create a solution concept for the subject matter being studied. Collaborative visual analytics includes analytical reasoning techniques, visual representations and interaction techniques, data representations and transformations techniques, and techniques that support production, presentation and dissemination of the concept solution.
The EBS process takes place in a time-pressured synthetic interactive analytical environment including multiple displays enabling interactive group collaboration and it facilitates iteratively non linear decision making processes through an active discourse supporting dynamic multi-stakeholder interactions.

The content creation workflow of the Ship Simulator game a case study

Pjotr van Schothorst
V-STEP B.V., Rotterdam, The Netherlands
pjotr@vstep.nl

Abstract

There is a world of difference between the academic theory of asset management and the daily practice of content creation and management in a game development company. Pjotr van Schothorst, co-founder and CTO of VSTEP, a Dutch developer of both entertainment games and training simulators, will explain the workflow VSTEP uses for creating their products. This is a process involving many steps and tools, with potential for improvement. Pjotr will discuss some of the changes VSTEP is working on, and the audience will be invited to come up with other improvements.

Computer Graphics Recipes for Endowing 3D Shapes with Semantics in Virtual Worlds

Michela Mortara Chiara Eva Catalano

CNR-IMATI Ge
Via de Marini 6, Genova, Italy
{michela.mortara, chiara.catalano}@ge.imati.cnr.it

Abstract

The massive use of 3D models increased also in the Gaming community the awareness of the benefits of a digital representation of objects involving not only geometric data but also their semantics. Encoding semantic information (that is, functionality or meaning) gives the opportunity of performing automatic reasoning on object's features, and interaction capabilities, behaviour and constraints. In a virtual environment scenario, associating 3D shapes with semantics may allow for smart search and retrieval of characters and objects, fast re-use of animation sequences and easy re-adaptation of content to new tasks and environments. Nonetheless, the process of semantic annotation should require the lowest effort from the designer side to be really feasible and applicable.

In this paper we are going to present the recent research outcomes of Computer Graphics devoted to the segmentation of 3D shapes into main components and to the identification of salient features, to the extraction and representation of knowledge related to 3D content, and to the association of such knowledge with the geometry of objects (to their components or as a whole). These, in fact, can be seen as the basic elements to build semi-automatic tools to support the semantic annotation of 3D shapes.

Keywords: Shape modelling, shape analysis, segmentation, annotation, 3D content, semantics

1 INTRODUCTION

The ongoing improvements in graphics hardware enable to create more and more complex and visually convincing virtual worlds for games and simulations. For the near future, the strongest expectancy in this field is steadily shifting from improved graphics and appearance towards more coherent gameplay, plausible realism and more natural interaction capabilities with objects and characters populating the virtual environment [24]. To this aim, the key ingredient is likely to be the capability of embedding the game elements with semantic data, ranging from high level characteristics of the whole environment or of its regions (e.g. climate, kind of vegetation), to object and character properties (e.g. weight, size, colour, personality) and their relationships (e.g. "contains", "owns"). Such semantic information will be crucial, on the one hand, to reuse effectively game elements in the design phase together with an efficient search, retrieval and coherent adaptation of game elements in new contexts; on the other hand, it will pave the road for more elaborate kinds of interaction within virtual worlds. The concept of *Smart Object* [16] goes in this direction: typically, an object representation takes into account only its appearance and consists of mere geometric data; conversely, its physical properties, roles, behaviour and services, which are crucial for defining its interaction capabilities, are usually not taken into consideration. A Smart Object can describe its own possible interactions (opposite to 'plain' objects that do not provide this information). The basic idea is that a virtual character does not need to know in advance how he/she can interact with every possible object, since he/she can "ask" the object which capabilities it has got.

To exploit fully the potential of semantic enrichment of 3D shapes in a game environment, as many properties of real-world objects as possible should be represented in their virtual counterparts; the more semantic data will be available, the easier and more complete will be an automatic reasoning on the objects, thus allowing more elaborate kinds of interaction with them and enhancing the realistic feeling of the environment. Nevertheless, the addition of

semantic data should not have a negative influence on the time and effort designers spend creating their virtual worlds. Therefore, in the gaming and simulation area as well as in many other application domains that are characterized by a massive use of 3D data there is an increasing awareness of the usefulness of embedding semantics into 3D models and consequently of the need for techniques to reduce the amount of manual annotation of digital 3D shapes.

A first step into the direction of automatic 3D annotation may be taken by means of the extraction and management of knowledge related to 3D digital shapes, which is matter of the Computer Graphics research; in particular, Shape analysis is necessary, which develops computational tools for reasoning on properties of the objects' shape, and is pivotal in a large number of applications.

To bring intelligence into 3D content, we need to find metaphors for describing objects that are, at some extent, equivalent to those that people employ in organising their view of the world. Humans easily perceive and describe objects using shape properties at a conceptual level, and in terms of the object's constituting parts by means of terms which are meaningful in some domain of knowledge. For example, a given model can be described as being a table, made of four cylindrical legs and an oval top.

Hence, a semantic description calls for *segmentation* algorithms which capture semantically relevant features in an automatic way. It is rather difficult, however, to evaluate the performance of the different methods with respect to their ability to segment shapes into meaningful parts. This is due to the fact that the majority of the methods used in computer graphics are not devised for detecting specific features within a specific context, as for example is the case of form-feature recognition in product modelling and manufacturing. Conversely, feature recognition for free-form shapes is challenging, especially when the semantics underlying the features are related to an intrinsically vague context, as with most natural shapes.

Shape segmentation and feature extraction are the computational tools needed to select the portion of interest in a model; selecting regions of interest in the manual annotation of 2D media is rather simple in terms of user interface: dragging a selection box or lasso tool over an image achieves the necessary functionality. The same does not hold for 3D media, or at least it is complicated by the data nature: parts might be out of reach for mouse interaction, and bounding a part can be rather complex. Most of the methods developed so far for shape analysis and segmentation do not directly provide any semantically relevant explicit description of the shape, but some of them at least provide a characterisation of the geometric and structural properties of shapes and their parts; then, shape segmentation is complemented by shape annotation, which is typically the means to document content with contextual knowledge, either manually or automatically.

In the following section we present some segmentation techniques developed in Computer Graphics that are able to identify semantically salient features of 3D objects targeting the recognition of different characteristics and that are suited to different types of shapes. We will show how these techniques may provide geometric and structural attributes of shape parts that, in very specific classes of objects, are sufficient to encode semantics in objects and their components automatically: an example is given in Section 3.1.

In general, only by combining different approaches we will be able to capture the relevant features of objects depending on the user needs, and annotate them with contextual knowledge in either an automatic or a computer-assisted way; in Section 3.2 we describe the ShapeAnnotator, a tool to annotate 3D shapes with domain specific knowledge formalised in terms of ontologies and is based on a library of semantics oriented segmentations.

Finally, in Section 4 we describe how the segmentation and semantic annotation of characters and objects may be successfully exploited in the gaming and simulation field, while Section 5 concludes the paper.

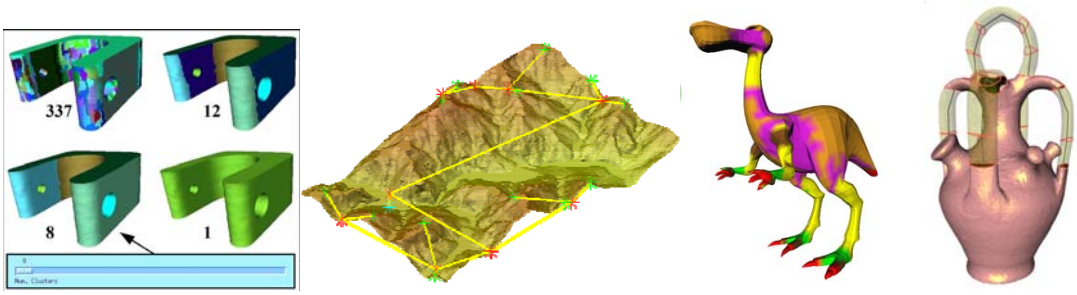
2 SEMANTICS-ORIENTED SEGMENTATIONS

Although plenty of segmentation methods have been proposed in the literature, the vast majority of them take into account only local geometric attributes of the shape to build up segments, disregarding the structural aspect and then the semantics of parts. Semantic properties are taken into account, at some extent, by cognitive theories supporting part based decompositions or minima rule based approaches. Part based decomposition techniques build on Biederman's theory of perception, which characterises an object as a compound of primitive basic parts (e.g. planes, spheres, cylinders, cubes) [7]. The second class of methods is based on the so-called *minima rule*, which suggests that we perceive relevant parts by focusing our attention on lines of concave discontinuity of the tangent plane [14]. For a recent survey of segmentation methods, we refer the reader to [22].

Our interest is in methods for shape analysis and segmentation able to capture a varied set of morphologically relevant features, possibly at different scales. Shape understanding is a very complex task indeed, and it is now

widely accepted that no single segmentation method is capable of capturing relevant features in a broad domain of shapes. Nonetheless, some segmentations exhibit such characteristics that make them particularly suitable for some specific shape classes. While detecting salient features, a segmentation algorithm usually computes specific geometric and topological descriptors of segments that could be used to annotate shape components not only with conceptual knowledge (e.g. "a leg of a table") but also with geometric attributes (e.g. "length=60 cm") needed for shape modelling and adaptation for content re-use in a virtual world.

In the following subsections we are going to briefly describe some of the semantics oriented segmentation techniques developed by the Shape Modelling Group at CNR-IMATI Genova [23]. These are: the Fitting Primitives, the Reeb graph, the Tailor and Plumber approaches (see Figure 1). We point out that these techniques are devoted to the segmentation of 3D models represented by manifold triangle meshes.



(a) (b) (c) (d)

Figure 1: Segmentations obtained by the fitting primitives (a), the Reeb graph (b), the Tailor (c) and the Plumber (d) methods on different shapes for which each method is best suited.

2.1 FITTING PRIMITIVES

In [4], a hierarchical face clustering algorithm for (combinatiorially manifold) triangle meshes is described based on their fitting to primitive shapes belonging to a predefined set, being able to recognise planes, cylinders, and spheres. The algorithm has the same structure of the hierarchical face clustering proposed in [10], in which only fitting planes are searched, but it has been designed to provide more flexibility, and can incorporate support for different fitting primitives and error metrics. This method is completely automatic and generates a binary tree of clusters, each of which fitted by one of the primitives employed.

Initially, each triangle represents a single cluster; at each iteration all the pairs of adjacent clusters are considered, and the one that can be better approximated by one of the primitives forms a new single cluster. The approximation error is evaluated using the same metric for all the primitives (here the standard L_2 metric), so that the user can choose which is the most suitable primitive to approximate the set of triangles in a cluster. Such a system proved to be extremely efficient and robust to noise. If the model is known to be made of a well defined set of primitives, as typical, for example, for mechanical objects, the algorithm may accept a plug-in for each of them in which the computation of both the fitting parameters and the error are implemented. Moreover, being a greedy method, the level of accuracy is somehow reflected by the cluster hierarchy which, once computed, may be interactively navigated by the user through a slider which sets the desired number of clusters or a threshold error, as shown in Figure 1(a). Finally, the algorithm calculates automatically the values of main parameters of the primitives, which are diameters and heights. The time complexity of this algorithm is, on average, O(N log N) being N the cardinality of mesh vertices.

In a virtual environment the Fitting Primitives method is best suited to segment crafted objects or architectural elements, such as the planes that form floors, walls, table tops; the computed numerical values of the primitive parameters can be recorded for possible rescaling and modifications of such objects.

2.2 REEB GRAPH

Morse theory [18] sets the foundations for associating, in the continuum, the topology of a given manifold M to the critical points of a smooth and real-valued function f defined on the manifold, called the mapping function. Based on results of Morse theory, it is possible to code the topology of a shape into a graph-like structure, called the Reeb graph of M with respect to f, which captures the evolution of the levels set of f on the shape. The nodes of the Reeb graph correspond to critical points of f which cause the level set to join and split, or to appear and disappear, while

arcs represent shape components where the topology of level set does not change. The Reeb graph naturally induces a decomposition of the shape into regions, bounded by level sets drawn at saddle points. Each region is characterised by a critical point of *f*, and a suitable choice of the mapping function *f* allows for detecting features with different geodesic properties. For instance, if the mapping function is the average geodesic distance, the Reeb Graph will properly characterise the protrusions and depressions of the shape in a pose invariant manner; the distance from a pre-defined set of high curvature points usually gives the most intuitive skeleton of a shape, while the height function is the most efficiently computed but has the drawback of depending on the object position in space. The time complexity of the graph construction as described in [6] is O(max(N log N, Mlog M)) where N, M is the cardinality of the mesh and of the constrained mesh respectively; the chosen mapping function must be evaluated on the surface in advance and must be taken into account as a pre-processing time.

Another intuitive application that may fit the annotation of virtual environments concerns digital terrain models using the height function, where the correspondence between segments and shape features is most evident (peaks, valleys, ridge lines are well characterised, see Figure 1 (b)).

2.3 TAILOR

The Tailor method [19] analyses a surface mesh at different scales through the assessment of the geometric and topological attributes of each surface point over neighbourhoods of increasing size. In each vertex a sphere is placed, whose radius corresponds to the scale at which the surface is being analysed. For very small radius values, the sphere intersects the mesh into one intersection curve, whose length is related to the curvature of the surface patch inside the sphere, and converges to the value of the Gaussian curvature in the centre of the sphere as the radius tends to zero. For larger radius values, the sphere may intersect the mesh in more than one curve, giving a hint on the topology of the neighbourhood for identifying protrusions or splitting regions. Other properties are then computed such as convexity/concavity of neighbourhoods. Finally, geometric and topological attributes concur to label each vertex with a high level morphological feature tag, such as *Tip* (disc-like topology, high curvature, convex), *Blend* (disc-like topology, negative Gaussian curvature), *Well* (tube-like topology, intersection curves of similar length, empty status), *Split* (more complex topology). The corresponding segmentation is simply achieved by clustering neighbouring vertices with homogeneous labels (see Figure 1 (c)). Computing the intersections between the mesh and a sphere placed in every vertex gives a computation time of $O(N^2)$ in the worst case, being N the vertex cardinality.

By its nature, the Tailor approach is best suited to characterise complex and articulated shapes that exhibit curvature features and branching parts, like plants and animals. Tailor is particularly useful to locate interesting landmarks on surfaces: in the case of human body models, landmarks are surface feature points on the body that can be used for a variety of purposes, such as the automatic computation of anatomical measures from anthropometric data, and are defined in several Standards (like CAESAR [8], H|Anim [15], ISO 7250). For example, landmarks like the sellion, navel, and armpit can be easily identified as the centres of *Dip* and *Blend* regions provided that the analysis is performed at the suitable scale (see Figure 2).

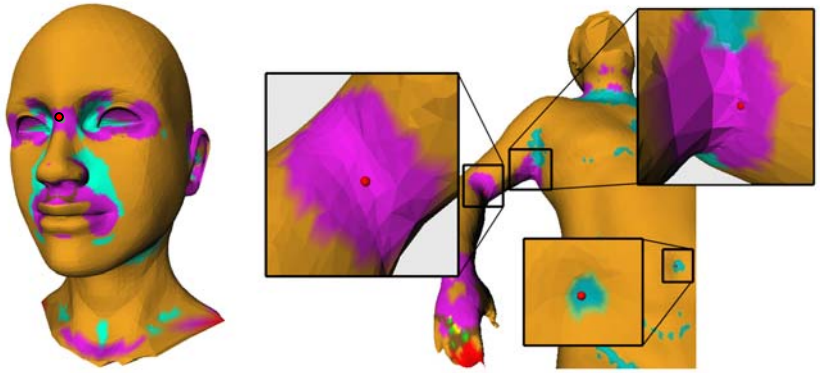


Figure 2: Identification of human body landmarks using the Tailor characterisation.

2.4 PLUMBER

Plumber [20] builds on the Tailor analysis and it is specialised for tubular parts. In fact, it defines a shape decomposition into connected components that are either *tubular features*, identified by regions which can be described as generalised cones or cylinders (e.g. handle-like and protrusion-like features, together with their concave counterparts), and *blob regions* identified by patches which connect tubular features. Like in Tailor, vertices belonging to tubular features are determined by both geometric and topological attributes of neighbourhoods of increasing size, defined by spheres centred in each vertex and having increasing radius. Therefore, the tube detection works in a multi-scale setting (i.e. using a fine-to-coarse strategy), starting with the extraction of small tubes first. The set of radii is automatically set by uniformly sampling the interval between the minimum edge length and the diagonal of the bounding box of the object; however, users can tune the level of details if they have some a priori information or if they search for features of given dimension. Additional parameters may be specified in order to either restrict the search to tubes of predefined section or allow smooth changes in size too. This segmentation is particularly suited for articulated objects, both natural and man-made, which typically exhibit elongated protrusions (see Figure 1 (d)), while it is inappropriate for unstructured, uniform shapes. Regarding the computation time, the dominant factor is the Tailor analysis that must be performed first, that is $O(N^2)$ where N is the number of mesh vertices. In the context of virtual environments Plumber has been successfully applied to enrich objects with grasping information [1]: in fact it was used to locate handles of a proper size in order to be grasped by an avatar, and to compute the appropriate hand position over it. Moreover, it perfectly fits the task of identifying human limbs in order to label/tag human body parts, as described in the following.

3 ANNOTATION OF 3D SHAPES

Shape segmentation and feature extraction are the computational tools needed to select the portion of interest in a model. As described above, some segmentation techniques are also able to provide geometric and structural attributes of shapes and/or shape parts that constitute some intrinsic geometric knowledge about the model. But not all the semantic information can simply be extracted from the geometry. Indeed knowledge about 3D content can be of different types:

- *Knowledge related to the geometry of the 3D shape*. While the descriptions of digital 3D media can vary according to the contexts, the geometry of the object remains the same and it is captured by a set of geometric and topological data that define the digital shape;
- *Knowledge related to the application domain in which the object is manipulated*. The application domain casts its rules on the way the 3D shape should be represented, processed, and interpreted; features are the key entities to describe the media content, and these are obviously dependent on the domain.
- *Knowledge related to the meaning of the object represented in the 3D model*. Digital models may represent objects that belong to a category of shapes, either in broad unrestricted domains (e.g. chairs, tables in the house furniture) or narrow specific domains (e.g. pockets in mechanical engineering).

Besides, the two most important aspects of an annotation framework involve what is annotated and how it is annotated. For the first aspect, it is important to understand whether the target is the whole object or any of its constituent parts. If the target is a constituent part, the annotation must be preceded by shape segmentation. For the second aspect, users may want to annotate 3D objects with free tags in applications inspired to social networking principles where users are undetermined. Oppositely, in collaborative working environments where annotations serve the purpose of enriching geometric data with specialised knowledge ontology-driven tags are more suitable. An *ontology* is a formal, explicit specification of a shared conceptualisation of a domain of knowledge, and expresses the structuring and modelling of that particular domain [10][11]. At the conceptual level, an ontology is defined by the specification of classes, with their attributes and properties that may also define relations between the classes.

Because encoding semantics in 3D shapes in general and 3D annotation in particular are a relatively new topic, few prototypes exist. In the following subsections, two approaches are described which support the part based annotation of 3D shapes and build on semantics oriented segmentations: the first is specifically tailored for the identification of conceptual parts of human body models and is completely automatic; the second can handle every kind of shapes and provide for the manual annotation of meaningful components with concepts formalised through domain ontologies.

3.1 AUTOMATIC IDENTIFICATION OF HUMAN BODY PARTS

In the context of human body models, while geometric attributes may vary from a model to another, the human body structure is well defined and the basic components are predominantly tubular (e.g. arms, legs, fingers, neck) and therefore can be easily identified by the Plumber segmentation. Indeed, [9] presents an automatic procedure for tagging human body parts based on the segmentation provided by *Plumber*. The labelling can be defined as a function $f : S \rightarrow L$ from the set of segments that may be either tubular or non-tubular (blob) parts to the set of labels that are in our case *trunk, arm, hand, palm, finger, fingertip, leg, foot, neck, head* and which correspond to conceptual tags.

The algorithm exploits the geometric attributes of parts, computed during the segmentation phase. For tubes these are the axis length and the maximum, minimum and average length of cross sections, while for blobs the volume is considered. The adjacency relations among segments and the a priory knowledge on human anatomy are exploited to define tagging rules of parts.

The automatic conceptual tagging of body parts has been successfully applied to the extraction of a skeleton for animation for Virtual Humans (VHs). An anatomically realistic skeleton has different characteristics in specific body parts: for instance, it is medial in limbs like the arms and legs; conversely, the spine lies straight under the skin of the back. The body part identification is therefore used to construct correctly each skeleton segment according to the specific body parts. The Tailor segmentation has been also applied to detect the correct location of body part specific joints and landmarks (see Figure 3).

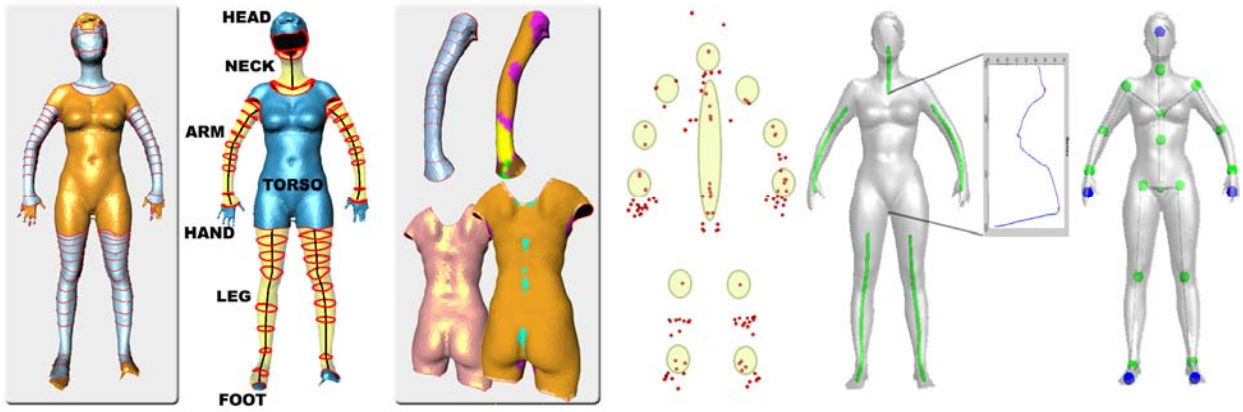


Figure 3: Pipeline of the approach in [9]: (a) the Plumber segmentation; (b) semantic annotation of a human body model; (c) Tailor characterization on specific body parts; (d) Candidate skeletal joints; (e) skeletal segments for limb parts; (f) the extracted animation skeleton.

3.2 THE SHAPE ANNOTATOR

The paradigm behind the ShapeAnnotator is based on the twofold assumption that an effective annotation must be related to a specific domain, and the definition of relevant shape feature must be related to the same domain.

Thus, the input of the ShapeAnnotator is constituted of (1) the shape to be annotated, (2) an ontology formalising the annotation domain, i.e. the contextual knowledge, and (3) an optional set of shape segmentation *plug-ins* to extract specific, domain dependent features. Plug-ins are optional because the ShapeAnnotator already includes several standard segmentation algorithms (including those described in Section 2), whose results can be composed and edited within a *multi-segmentation* to define non-trivial features. In many cases, in fact, shape features are not sharply defined in terms of their boundary; if the annotation domain describes a head and a torso to be adjacent parts of a human body, for example, the neck should be considered part of both. In general, features may overlap and they do not necessarily form a partitioning of the whole (i.e. some parts may remain undefined as they do not have any particular meaning in the context addressed).

Once segmentation algorithms have been run to properly define interesting features, the next step consists of selecting each such feature and tagging it with a concept of the ontology, that is creating instances of the ontology. The domain ontology coupled with the instances form a knowledge base in which the semantics is connected to the geometry. To this aim the ShapeAnnotator provides an integrated ontology browser to seek and select the proper concept by navigating across ontological relations, and to create an instance for the resulting knowledge base.
The output of the annotation process is a pair of files that encode (see Figure 4):

- The geometry of the shape and its interesting features;
- The set of instances describing the features.

.

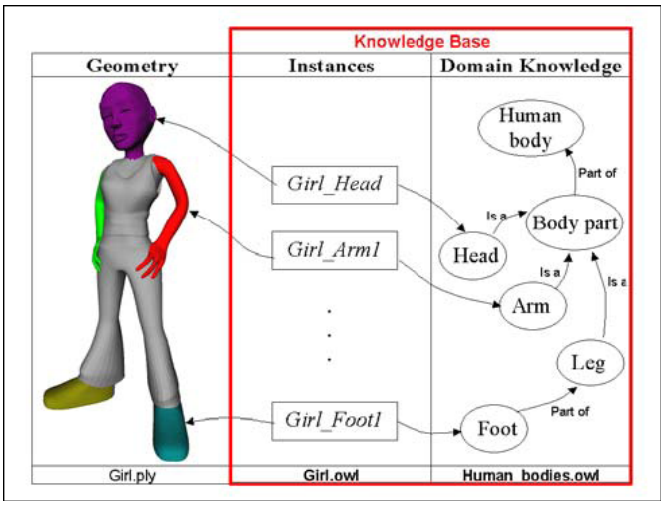


Figure 4: *girl.owl* encodes instances of concepts formalized in *Human_bodies.owl* and, at the same time, points to the corresponding geometry in *girl.ply*.

4 APPLICATION SCENARIOS

The automatic identification of human body parts described in Section 3.1 permits to optimise the skeleton construction and landmark location of Virtual Humans (VHs), which give the possibility to create efficiently new animation sequences. While the part identification is a means for a more efficient enrichment of 3D media for further processing or reuse, the ShapeAnnotator even permits to share and retrieve 3D resources and their subpart by simple text based queries according to relevant concepts. In fact, as we have seen, the ShapeAnnotator allows the user to tag the different parts of a segmented shape according to one or more ontologies of interest. In this way, a resource and its subparts can be retrieved in a shape repository according to different conceptualisations, and then reused and enriched in another context decreasing the number of manual operations typical of the design and animation pipeline.

An example of such an application scenario in the gaming field can be outlined with the tools offered by the Digital Shape Workbench (DSW) developed in the framework of the AIM@SHAPE [2] project. The DSW is a common infrastructure for integrating, combining, adapting, enhancing software tools and shapes with the goal of speeding up research in digital shapes. In addition to a shape and a tool repository, it includes two Common Ontologies (COs), one for shapes and the other one for shape processing tools, which allow for a generic (i.e. non-domain related) specification of respectively shape representations and shape processing tools. The COs have been extended with three ontologies, which address the need of particular shape application domains, i.e. Product Design, Virtual Humans, and Shape Acquisition and Processing. In particular, the *Virtual Human Ontology (VHO)* is related to the description of Virtual Humans to simplify the composition of avatars by non-experts and to facilitate sharing of useful information including morphological data, behaviour, and animation [13] (see Figure 5); for example, it is possible to distinguish between *motion capture* and keyframe animation sequences, and to attribute a particular emotional state to a virtual character.

The VHO can be helpful in those applications aiming at making virtual simulations involving humans, such as the population of Virtual Environments, where one of the main challenges is to create a large diversity of human characters to fulfil the demand of a large number of users. A typical scenario may include an advanced search of

avatars and smart objects (whether full or partial bodies) in a repository -in the Shape repository in this case- with several specific characteristics. They can be simple features like age, gender, weight, but also complex as related to animations with specific conditions, e.g. a mature woman running. If the character is equipped by a skeletal structure, animations exploiting such information (e.g. node position and attributes) can be applied.

In the modelling phase, a designer could search for VH shapes (whether full or partial bodies) and use them to model new VHs, improve or reconstruct existing ones; an expert modeller is able to even retrieve geometric and structural features (landmarks, topological graphs, and so on). Among the users researchers working on algorithms for recognising features on a shape representing a virtual/real human may be included. Data would be used on ergonomics studies, computer vision algorithms, etc.

Since the VHO includes also garment characterisations, it could help, for example, with the management of an on-line shop to build custom 3D mannequin and select and fit clothes and accessories, with real-time pre-visualisation. It would help to decide on the most suitable clothes and accessories for a specific VH by matching the corresponding landmarks.

Another application scenario where the importance of encoding knowledge in a virtual environment is crucial is the assistive living. Here the development of a framework which supports the design and validation of domestic environments with ICT based accessibility support for people with disabilities and functional limitations may benefit of a virtualisation both of the environment and of users of different profiles. This would not simply provide a realistic visualisation but a realistic simulation of places, people and danger conditions when associating semantics to the geometric representation of the elements to be virtualised: for the virtual users, parametrised body models and models of behaviour and motion, according to different users' profiles, can be implemented; for the domestic environment, the algorithms described in this paper could be exploited for the detection of functional features in the objects and for the semantic annotation of the habitat.

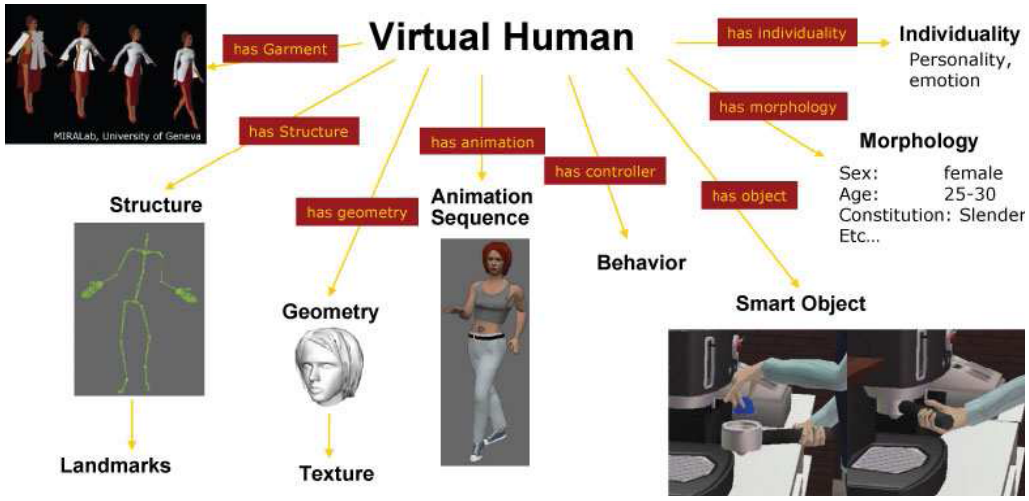


Figure 5: General scheme of the Virtual Human Ontology

5 CONCLUSION

This paper discusses the key aspects of endowing 3D shapes with semantics and shows how the explicit encoding of knowledge related to virtual objects may become crucial in applications involving virtual worlds.

Nevertheless, the annotation process has been, until recently, completely manual, apart from low level geometric attributes that may be automatically computed and associated to a 3D model. This explains the increasing demand for tools that support 3D content annotation in various fields and that can manage different kinds of knowledge related to a 3D shape: not only knowledge about its geometry is required but also about its meaning and the context of use.

To make geometric knowledge explicit we propose using shape segmentation techniques, able to identify the most interesting features of a digital model and supporting a structural description of the shape. As an example, we showed how, in restricted domains like the one of human body models, specific segmentations allow for the identification and labelling of shapes and their parts with conceptual tags.

Then, we introduced the ShapeAnnotator, a tool addressing the part based semantic annotation of shapes through the adoption of ontologies. In its current version, the ShapeAnnotator has minimal inference capabilities to provide a flexible browsing of the ontology, but future developments are targeted to this aspect in order to automate (partially) the whole annotation pipeline.

We believe that embedding 3D shapes with semantics will highly impact on the process of designing virtual worlds for gaming and simulations: in fact, as depicted in the application scenarios described, such enriched 3D models can more easily be indexed, searched and retrieved and efficiently re-used by coherently re-adapting the content to a new environment. Moreover, earlier developments demonstrate that 3D shapes endowed with semantic information about their features and functionalities will make interactions among virtual actors and objects more and more complete and credible, thus increasing the realistic feeling of synthetic environments.

ACKNOWLEDGEMENTS

This work is partly supported by the FOCUS-K3D EU CA (Contract N. ICT-2007.4.2). Thanks are due to Michela Spagnuolo, Bianca Falcidieno and all the members of the Shape Modelling Group of the CNR IMATI-GE for their helpful discussions and suggestions. We also thank the partners of the AIM@SHAPE project who actively collaborated to the development of the DSW and the VHO described in this paper.

REFERENCES

[1] Abaci, T., Mortara, M., Patanè, G., Spagnuolo, M., Vexo, F., Thalmann, D. (2005). Bridging Geometry and Semantics for Object Manipulation and Grasping. In SVE05: *Proceedings of Workshop towards Semantic Virtual Environments* (Villars, Switzerland, 16-18 March 2005)., pages 110-119., 2005

[2] AIM@SHAPE: Advanced and Innovative Models And Tools for the development of Semantic-based systems for Handling, Acquiring, and Processing knowledge Embedded in multidimensional digital objects, European Network of Excellence, Key Action: 2.3.1.7 Semantic-based knowledge systems, VI Framework, Contract IST 506766, URL: *http://www.aim-at-shape.net*; *http://dsw.aimatshape.net*.

[3] Attene M., Robbiano F., Spagnuolo M., Falcidieno B. (2007). Part-based annotation of virtual 3D shapes. In *Proceedings of Cyberworlds' 07*, Special Session on NASAGEM workshop. Silver Spring, MD: IEEE Computer Society Press. pages 427–36, 2007.

[4] Attene M., Falcidieno B., Spagnuolo M. (2006). Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22, pages 181–193, 2006.

[5] Attene M., Katz S., Mortara M., Patané G., Spagnuolo M., Tal A. (2006). Mesh segmentation-a comparative study. In *SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006*, IEEE Computer Society, 2006.

[6] Biasotti S., Giorgi D., Spagnuolo M., and Falcidieno B. (2008). Reeb graphs for shape analysis and applications. *Theor. Comput. Sci.*, 392(1-3), pages. 5–22, 2008.

[7] Biederman I. (1987). Recognition-by-components: a theory of human image understanding. *Psychological Review,* 94(2), pages 115–47, 1987.

[8] Civilian American and European Surface Anthropometry Resource Project CAESAR http://store.sae.org/caesar/

[9] Dellas, F., Moccozet, L., Magnenat-Thalmann, N., Patané, G., Mortara, M., Spagnuolo, M., Falcidieno, B. (2007). Knowledge-based extraction of control skeleton for animation. In *Proceedings of the IEEE international Conference on Shape Modeling and Applications 2007* (June 13-15, 2007). IEEE Computer Society, Washington, DC, pages 51-60, 2007.

[10] Garland M., Willmott A., Heckbert P.S. (2001). Hierarchical face clustering on polygonal surfaces. In *I3D'01: Proceedings of the 2001 symposium on interactive 3D graphics*. New York, NY, USA: ACM; pages 49–58, 2001.

[11] T. R. Gruber (1993). Towards principles for the design of ontologies used for knowledge sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.

[12] Guarino, N. (1998). Formal ontology and information systems. *In N. Guarino, editor, Formal Ontology in Information Systems*, IOS Press, Amsterdam, pages 3–18, 1998.

[13] Gutierrez, M., Garcıa-Rojas, A., Thalmann, D., Vexo, F., Moccozet, L., Magnenat-Thalmann, N., Mortara, M., Spagnuolo, M. (2007). An Ontology of Virtual Humans: incorporating semantics into human shapes. *The Visual Computer*, vol.23 (3), Springer, pages. 207-218, 2007.

[14] Hoffman D.D., Richards W.A. (1994). Parts of recognition. *Cognition*;18 (1–3), pages 65–96, 1984.

[15] Humanoid Animation Working Group H|Anim http://www.h-anim.org/

[16] Kallmann, M., Thalmann, D. (1998). Modeling objects for interaction tasks. In *Proceedings of the 9th Eurographics Workshop on Animation and Simulation* (EGCAS), pages 73–86, 1998.

[17] Kallmann, M., Aubel, A., Abaci, T., and Thalmann, D. (2008). Planning collision-free reaching motions for interactive object manipulation and grasping. In *ACM SIGGRAPH 2008 Classes* (Los Angeles, California, August 11-15, 2008). SIGGRAPH, ACM, New York, NY, pages 1-11, 2008.

[18] Milnor J. (1963). Morse theory. *Princeton University Press, Princeton, NY*, 1963.

[19] Mortara M., Patane` G., Spagnuolo M., Falcidieno B., Rossignac J. (2003). Blowing bubbles for multi-scale analysis and decomposition of triangle meshes. *Algorithmica*; 38, pages 227–48, 2003.

[20] Mortara M., Patane` G., Spagnuolo M., Falcidieno B., Rossignac J. (2004). Plumber: a method for a multi-scale decomposition of 3D shapes into tubular primitives and bodies. In *SM'04: proceedings of the ninth ACM symposium on solid modeling and applications*. EurographicsAssociation; pages 339–344, 2004.

[21] Petridis K., Anastasopulos D., Saathoff C., Timmermann N., Kompatsiaris I., Staab S. (2006). M-OntoMat-Annotizer: Image annotation. linking ontologies and multimedia low-level features. In *KES 2006 - 10th Intnl. Conf. on Knowledge Based, Intelligent Information and Engineering Systems* October 2006, .

[22] Shamir A. (2006). Segmentation and shape extraction of 3D boundary meshes. In *Eurographics STAR report*; 2006.

[23] Shape Modelling Group – CNR IMATI Ge home page http://www.ge.imati.cnr.it/ima/smg/home.html

[24] Tutenel, T., Bidarra, R., Smelik, R.M., and Kraker, K.J. (2008). The role of semantics in games and simulations. *Comput. Entertain.* 6, 4 (Dec. 2008), pages 57:1-35, 2008.

Rule-based layout solving and its application to procedural interior generation*

Tim Tutenel and Rafael Bidarra
Delft University of Technology
Delft, The Netherlands
timt@graphics.tudelft.nl, r.bidarra@ewi.tudelft.nl

Ruben M. Smelik and Klaas Jan de Kraker
TNO Defence, Security and Safety
The Hague, The Netherlands
{ruben.smelik|klaas_jan.dekraker}@tno.nl

Abstract

Due to the recent advancement in procedural generation techniques, games are presenting players with ever growing cities and terrains to explore. However most sandbox-style games situated in cities, do not allow players to wander into buildings. In past research, space planning techniques have already been utilized to generate suitable layouts for both building floor plans and room layouts. We introduce a novel rule-based layout solving approach, especially suited for use in conjunction with procedural generation methods. We show how this solving approach can be used for procedural generation by providing the solver with a user-defined plan. In this plan, users can specify objects to be placed as instances of classes, which in turn contain rules about how instances should be placed. This approach gives us the opportunity to use our generic solver in different procedural generation scenarios. In this paper, we will illustrate mainly with interior generation examples.

Keywords: Procedural generation; constraint solving

1 INTRODUCTION

Many recent games that play in an urban setting feature huge cities, e.g. "Grand Theft Auto IV" (2008) or "Assassin's Creed" (2007). The player is however limited to entering only a handful of the many hundreds of buildings. It would obviously take too long for designers to model by hand the interiors of all these buildings with rooms and furniture, but with current procedural generation techniques, the interiors of these buildings could have been generated automatically.

This would not only increase the perceived realism of the game, more importantly, it would allow for new gameplay. The player could break into buildings to hide from enemies or to find food or other items like first-aid kits. It could also make chase sequences more interesting, and give players even more to explore in sandbox-style games.

Clearly, the difference in quality between hand-designed and procedurally generated content in the game world should not be too noticeable. In comparison to manually designed content, current procedural content can look dull and repetitive. However, there are several characteristics of building interiors that make them a suitable candidate for automated techniques. For example, often the interiors in a common house roughly follow the same structure. In a kitchen, we often see cabinets and counter being placed against the walls, with a table or perhaps an extension to the counter as an island in the middle of the room. Similarly, in a living room the couches are often placed around a small table and oriented towards the television set. Many such observations could be translated into rules and procedures to automatically generate these interior spaces.

Moreover, research in solving of space layout problems aimed at room interiors has already generated promising results. Several methods of solving different kinds of layout constraints among objects inside a room have been proposed, as we will show in the next section. We developed a rule-based layout solver, which is especially suited for procedural methods: based on a plan or a procedure, objects (e.g. furniture) are fed to the solver, which tries to fit them based on a set of rules defined for those objects and the ones that are already placed. We also define object features to steer the layout (e.g. areas around an object that should remain empty) or to link them to other objects. Furthermore, our approach was designed to be integrated with a comprehensive semantic class library which is explained in Tutenel, et al. (2009). In this paper, however, we will only briefly describe how these semantic classes

* This research has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie).

are used in our method. A more detailed discussion about how semantics can play an important role in the design of game worlds can be found in Tutenel, et al. (2008).

We first present the general idea of our rule-based layout solver in Section 3, after which we go into more detail about the layout planner which feeds the solver in Section 4. Finally, in Section 5, we show several examples where the solver is used to generate different layout problems.

2 RELATED WORK

Research on procedural generation of content suitable for game worlds has focused on many different aspects, including a variety of techniques that generate height maps for terrains or models for vegetation. A recent survey of procedural methods can be found in Smelik, et al. (2009).

Because of the characteristics of room interiors, we focus here on buildings and floor plan generation examples. We mentioned before that, in almost every room that has a specific function, patterns are visible. Many of these patterns are also hierarchical in nature: chairs are placed around the table, plates on top of the table and the cutlery, in turn, is placed next to the plates. A similar observation is made with respect to the decomposition of buildings and their facades. A wall contains a door and windows and those windows consist of a windowsill, the frame of the window and the glass inside that frame. Many algorithms based on this kind of decomposition supply shape grammars to generate the buildings and facades. See for example, the work of Wonka, et al. (2003), Yong, et al. (2004), Müller, Vereenooghe, et al. (2006), Müller, Wonka, et al. (2006) and Larive and Gaildrat (2006).

In floor plan generation methods we see the notion of shape grammars come up as well. In Rau-Chaplin, et al. (1996) and Rau-Chaplin and Smedley (1997), a shape grammar is presented to layout the different areas in a house. When this process is finished, these areas are assigned a function. A different method to automatically creating floor plans is proposed in Martin (2006). First a graph is generated in which every node represents a room and every edge corresponds to a connection between rooms. Next, these nodes are given a center location and the rooms are formed using growth rules.

An important advantage of procedurally generating building interiors is shown in Hahn, et al. (2006). Their research focused on generating only the rooms that are visible from the current viewpoint. This is obviously an efficient way of handling large buildings with many different rooms (e.g. office sky scrapers). To maintain changes made in the world, all changes are tracked and stored. When a room is removed from memory at one point, and is regenerated later on, the stored changes are again applied to the regenerated room.

Layout solving based on object rules is also applied in manual scene editing systems. In Xu, et al. (2002), objects contain rules describing which type of objects their surface supports. For example, food, plates or cups can be supported by a table or a counter. Smith, et al. (2001) use similar links, but they apply them to areas. They define offer and binding areas between objects, e.g. the area underneath a table can be an offer area that is linked to the binding area of a chair. In the WordsEye system, described in Coyne and Sproat (2001), natural language sentences are transformed to a list of objects with a set of constraints, based on which a scene is generated.

Declarative modeling of virtual environments (see Gaildrat (2007), Le Roux, et al. (2001) and Le Roux, et al. (2004)) combines constraints and semantic knowledge in the form of implicit constraints, to help the user generate a scene. In the description phase, the designer can express how a scene should look like. These descriptions are translated into constraints that are then fed to some constraint solver. Our layout solver uses a similar workflow: it uses rules defined for objects to come up with a set of constraints for to the solver.

A number of constraint solving techniques have already been researched to create room layouts in the form of space planning problems. Charman (1993) gives an overview of how existing constraint solving techniques that are not specifically focused on space planning can be applied to these problems. He discusses the efficiency of the solving techniques and compares several space planners. Many improvements for the discussed constraint solving techniques have been researched in the years following this study, so the results concerning the efficiency are no longer relevant. The discussed techniques, with their recent improvements, are however still applicable to layout solving. The planner he proposed, steered by the conclusions from his study, works with axis-aligned 2D rectangles with variable position, orientation and dimensions. Users can express geometric constraints on these parameters, which can be combined with logical and numerical operators. While our approach has similarities in the actual solving method, i.e. they both express the possible positions of an object as a union of areas; we propose different ways of expressing rules between the objects. While direct geometric constraints can still be expressed in our system, we allow more freedom to what these constraints can relate to and allow more indirect approaches through the use of features which will be explained in Subsection 3.1.

In the past, several space planning methods were developed using constraint logic programming (CLP). Even some decades ago, this approach was researched (see: Pfefferkorn (1975) and Honda and Mizoguchi (1995)). A

more recent system that used CLP was created by Calderon, et al. (2003). It is a framework that generates a number of different layout solutions for a number of objects, through which the user of the framework can interactively find desirable solutions. The rules for the objects are all expressed in predicate logic statements. This gives the opportunity to provide users with more or less natural language-like rules. In our approach we tend to work to a more visual solution, which we think might be more suitable for designers of game worlds.

3 A RULE-BASED LAYOUT SOLVING APPROACH

The main idea of our approach can be summarized as follows: given a starting layout, find the possible locations of a new object, based on a set of rules for that new object and objects in the layout. The relationships between objects can be defined in two ways: the explicit way, for example defining that the sofa needs to face the TV and that it should be no more than five meters away from it; and the implicit way, through the use of features, which we will explain next. An important aspect of our approach is the use of hierarchical blocks in the solving process. When placing a table with some chairs around it and a couple of plates on top, these objects are combined and treated as one block. This way, the solving process is made more efficient.

3.1 FEATURE-BASED CLASS REPRESENTATIONS

In our solving approach, each class can define a geometric representation valid for all instances of that class. This geometric representation consists of a number of so-called object features, which are 3D shapes containing a tag. These tags can refer to specific feature types. For a feature type, rules are defined about which features can and cannot overlap with them. For example, the *OffLimit* features cannot overlap any other features, e.g. the solid parts of objects (usually the entire bounding box). The *Clearance* feature denotes that this area of the object needs to be kept free e.g. for a person to walk or to use the object like the area in front of a closet. These features cannot overlap with other features, except other *Clearance* features.

As an example, a geometric representation for a table consists of five *OffLimit* features, corresponding to the four legs and the table top. The top feature can be assigned a *TableTop* tag. This way we can define that, for example, objects like a cup or a plate should be placed on a feature with the *TableTop* tag. The geometric representation is defined for a specific class, in this example the *Table* class. The shapes of the features are defined relative to the object, e.g. on top, or to the left of the object. This allows designers to link 3D models of different sized tables to the same *Table* class, and the five features will automatically be added to the models. It is however necessary that the models are uniformly oriented to enable the system to handle the relative descriptions of the feature shapes; e.g. when defined that we need a *Clearance* feature *in front of* every closet, the system will not generate the features correctly if a model with the wrong orientation is used.

Another example of how one can use features to position objects is the common case that an object should be placed against the wall. When creating the starting state for laying out a room, *Wall* features are added to the solver on the room's walls. By expressing that an object needs to be located with its back against such a feature, this rule can be enforced.

3.2 CLASS RELATIONSHIPS

In the previous section, we explained how we can implicitly add relationships between objects with the help of features. More detailed relationships can be defined in the rules that are used by the solving mechanism. These rules can be specified in two ways. They can be associated with a class, which will add the rule to every instance of that class, but the rule can also be defined in the layout planner (see Section 4), which gives the opportunity to define rules to objects that are not generally applicable. For example, a chair in a waiting room is generally placed against the wall, so this rule is specified only in the waiting room plan.

When creating the rules, one can express object relationships in a number of ways. First, we can create a direct link with an already placed object in the layout. This is only useful when creating a layout plan (see Section 4), because in a general class rule you cannot be sure which objects are already placed. The second way of expressing a relationship, which can be used in a general definition, is by linking an object to objects belonging to a specific class. We could, for example, define as a rule of the *Sofa* class, that when there is an instance of the *TV* class present in the room, a sofa should be facing that TV.

It is clear that in this approach a hierarchical relationship is created between the objects in a layout. For a designer, most of these hierarchies are clear, so he or she can make use of this knowledge by creating sub-plans. An interesting example of this is an office desk setup. A number of objects like an office chair, a telephone or a computer are all positioned relative to the desk. It is therefore practical to define a custom plan for creating such a

desk layout. Not only is it easier to reuse already created plans, it also provides an opportunity to speed up the solving process. Instead of placing the desk and all its related objects immediately, we can first place a *Desk* feature in the layout. This *Desk* feature could contain sub-features to guarantee clearance areas for example; still it will definitely be faster than individually placing the desk and all its related objects. After a suitable layout for an office space is generated using these *Desk* features, the features can then be replaced by all its related objects. This has the advantage that the placement of these related objects has become a sub-problem, for which it is faster to generate a solution.

3.3 SOLVING MECHANISM

In this section we describe how the solving approach works. First, we find all possible locations for a new object, based on the *ground* type of the object, its features, and the features of the already placed objects in the current layout. This *ground* type is the feature type on which the object can be placed. This could be a *TableTop* for a cup, a *Floor* for a table or a *Counter* for a kitchen sink. All these features have a shape which makes up the basic location for the new object. Above, we mentioned that *OffLimit* and *Clearance* features have a special meaning in this phase of the solving procedure. Based on the features of these types, unwanted overlaps are trimmed from the found locations. To allow for different orientations of the object, we perform this procedure for a discrete set of angles. In our system we use Minkowski addition to calculate the unwanted areas: when a feature of the new object should not overlap with an already placed feature, the Minkowski sum of the already placed feature and the new feature is trimmed from the possible locations. Based on the possible locations defined by the rules connected to the object, the list of possible locations is further refined. This mechanism is further illustrated here in pseudo code. Based on the input of a new object and the current layout, this algorithm creates a list of possible locations for the new object:

```
//--- Creating the list of possible locations of the new object based on the object's
//--- features and the features already placed in the current layout
//--- We start off with locations of the ground type features available in the layout
possibleLocList = currentLayout.GetFeatureLocationsOfType(newObject.GroundType)

//--- Now we prune this list of possible locations based on the overlap rules of the
//--- features in the new object and the already placed features in the current layout
for each objFeature in newObject
{
   //--- Each feature in the current layout that cannot overlap with the currently assessed
   //--- feature is subtracted from the possibleLocList
   for each layoutFeature in currentLayout
   {
      if ( ! objectFeature.OverlapAllowedWith(layoutFeature) )
      {
         //--- Using the Minkowski Sum, we create an area that contains all locations for
         //--- which the object feature would overlap with the layout feature
         illegalAreaAroundFeature = (layoutFeature.Shape).MinkowksiSum(objFeature.Shape)

         //--- Now we subtract this area from the list of possible locations
         possibleLocList = possibleLocList.Minus(illegalAreaAroundFeature)
      }
   }
}

//--- The possible location list is intersected with the possible locations based on each
//--- individual rule for the new object
for each rule in newObject
{
   ruleLocationList = rule.CreatePossibleLocations()
   possibleLocList = possibleLocList.Intersection(ruleLocationList)
}
```

We have two kinds of rules that can be handled by the solver. The *area*-based rules define a possible placement area for an object. This could be an area next to a feature or an object, on top of an object, etc. These are handled first by the solver and the intersection of these areas and the already found areas are now the provisional possible locations for the new object. Next, the so-called *grid*-based rules are handled. The current list of possible areas is cut into a grid of smaller areas, of which the size can be set inside the rules (when placing a table in a room, we can use larger grid sizes than when placing a fork on the table). For the center point of these areas a list of geometric

constraints can be evaluated, ranging from a required distance to a certain object or feature, whether another object is visible from that area, etc. The areas for which these constraints do not hold are discarded.

This black-or-white approach is not always desirable. We want to be able to define that an object of a particular class attracts or detracts objects of another class. For this we use attractors and detractors. These assign weights for the possible placement areas found previously. These weights will deem some locations as unlikely but not completely invalid for a specific object. In the final step of the process, we first pick an area based on the weights and subsequently we pick a random location within that area.

The same approach can be applied when designing a room by hand. When the designer wants to add a new object in a layout, the possible locations can be shown as a guide. Another possibility is to snap to a valid position closest to where the designer dragged the new object. This makes our solving approach suitable for both manual and automatic layout systems.

Performance is always an important issue for any solving approach. However, a generic approach will not be able to take advantage of many optimizations available for more specialized solving methods. In the algorithm above, it becomes clear that the number of features available in the layout could create an important bottleneck for the performance. Every time an object is added to the layout, all its features are added as well and the pruning of possible object locations based on the features in the layout will take longer and longer. The worst-case complexity, i.e. when every feature cannot overlap with every other feature in the layout, is near $O(n^2)$ with *n* the number of features in the new objects that need to be added to a layout. This is the reason why the hierarchic subdivision of the scene is very important, since the new object will only take into account the already placed features in the sub layout that is considered at that moment, so *n* will be the number of new objects added to the sub layout which can be considerably smaller. An optimized scene management system that stores the features of the layout can obviously further improve the performance of our approach. Our current implementation of this approach does not include mechanisms to detect contradicting rules. However, because of the step by step approach, it is quite easy to give the user insight in how a specific rule influences the possible locations for an object in an example. The system could show what areas are deemed valid or invalid by each rule, after which these rules could be adjusted by the user.

4 LAYOUT PLANNER

The job of the layout solver, discussed in the previous section, is to place an object in a layout, making sure the rules defined for that object in the class description hold. The layout planner submits objects to the solver one by one. A planner works based on a procedure: a list of rules that need to be executed in order. Examples for such rules might be: “place X instances of class Y”, or “place as many objects of class Z as possible”. This planner can also contain elements like if-then-else statements or loops: “keep adding cupboards until the total amount of storage space exceeds 1.3 cubic meters”, or “if context is “dinner” place 5 plates on a table in front of a chair and stack 10 plates in storage features, else stack 15 plates in storage features”. We can extend the previous example and define that in a context “after dinner” the plates should be placed on the sink. Note that currently, context in our solver is simply a list of tags that describe the current context, e.g. {“weekend”, “after dinner”, “near Christmas”…}. Based on these rules the planner will feed the layout problem solver. The problem solver, in turn, will make sure the rules defined in the classes are respected and therefore a valid solution will be generated.

An important rule that is available in the planner is the backtracking rule. When a point in a layout process is reached where an important object cannot be placed anymore, this could be solved by backtracking and choosing a different location for previously placed objects. At their old location, these objects might have prevented the placement of the new object.

When adding a new object to the layout, the corresponding features are placed accordingly, but the planner also allows the designer to directly add features to the layout. One of the feature types that is useful in a planner step is the *Area* feature. It is used to create a rough provisional layout, which in later steps can be filled with specific objects. This way, the *Area* features serve as kind of placeholders for the eventual objects. We could, for example, start our living room plan by first placing a sitting area and a dining area. Later, when we add a dining table to the layout, a rule is added that it should be placed inside the previously marked dining area. When placing a new *Area* feature, it behaves like an *OffLimit* feature, i.e. it may not overlap with any other feature and therefore guarantees the new feature area is empty. But when the feature is already placed, overlap with all other features *is* allowed with the exception of other *Area* features, and so the feature areas can be filled with the appropriate objects.

To edit and test the layout plans, we created a tool that allows designers to see how changes he or she makes to the plan affect the result. Because this result is obviously very dependent on the situation, we immediately show the effect of the changes on multiple examples (see Figure 1). Due to the nature of the problem designers will never be able to check all possible solutions, but by checking some solutions under different circumstances, major problems or unwanted behavior will quickly be noticed.

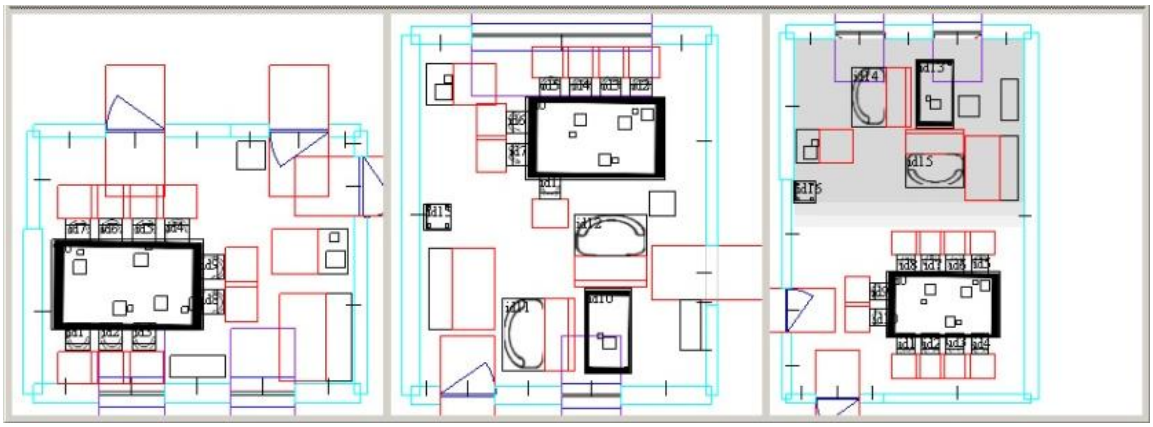


Figure 1: The impact of changes in the plan is immediately shown in multiple example situations.

5 EXAMPLES

When testing the layout solver, we mainly focused on room interiors. We will now show some results from a couple of the example scenarios we used. In our living room example, we used a plan that involved placing a dining area with a table and some chairs, a sitting area with a TV, a coffee table and two sofas, some furniture like a lamp and some cupboards and finally some extra objects like vases, plates and cups. This plan involved many of the example rules we explained in previous chapters, as well as some object relationships through linked features. It also contains some backtracking, e.g. when the coffee table cannot be placed in front of the TV. An example of such a layout can be seen in Figure 2. This is a moderately complex layout plan with, in total, 30 placed objects of 13 different classes each with between one and three rules, and it takes on average 152 milliseconds (on an Intel Core 2.4 GHz processor) to create a valid layout. This makes if fast enough to be used to generate and populate rooms at runtime. The solver could, in the background, generate the appropriate layout for the rooms closest to the player's current position. This approach will also result in a decrease of disk space necessary to store a game world, which can be important for online games, since downloading new areas of an online world can sometimes take a considerable amount of time and bandwidth and therefore more stress on the server.

We created a factory example to show how feature areas can be used to create a rough layout by positioning the (invisible) feature areas, and then positioning corresponding objects inside those areas. In the first steps of the plan we add some feature areas, each with their preferred size and rules. In this case we have a storage area, a locker area and a vehicle area. The locker area needs to be placed against the wall and the vehicle area close to both the entry gate and the storage area. Then we add the actual objects, each with their class rules, plus the extra rule that they need to be located in the correct feature area.

Using a similar approach we can use our solver to generate floor plans for a building. In our example in Figure 4, we used a fixed shape of the building. The bottom wall is next to the street. The rooms are represented as feature areas in the solver and are shown as rectangles in the images. We added some rules to the room areas, e.g. the hall should connect to the street-side wall and the living room should be adjacent to the kitchen. To *grow* the boxes to form the actual rooms, we used a separate method that expanded the rooms (based on priority) until it reached an outside wall or another room.

Figure 2: An example of a living room layout which took on average 152ms to generate layout with 30 objects of 13 classes.

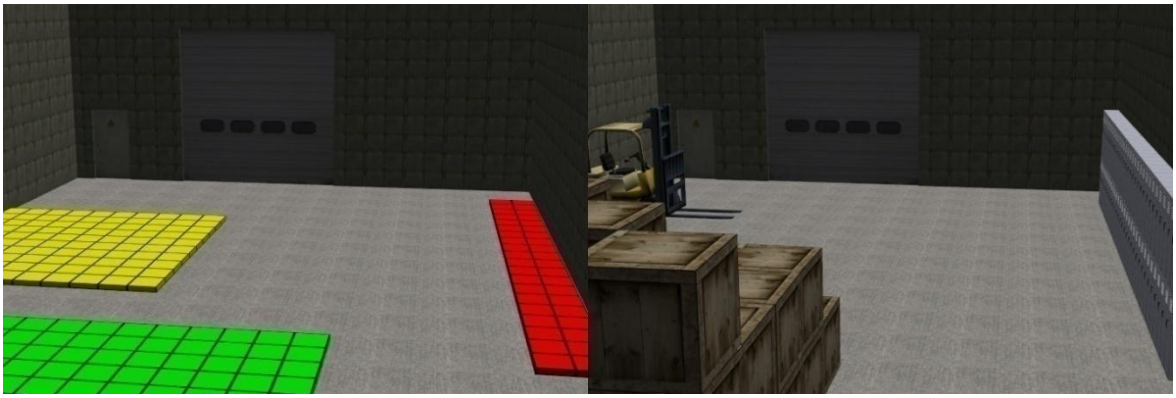


Figure 3: An example of a factory floor. First the different areas are positioned (left), and after that these areas are filled with the preferred objects (right). The scene includes over 100 objects and took 431ms to generate.

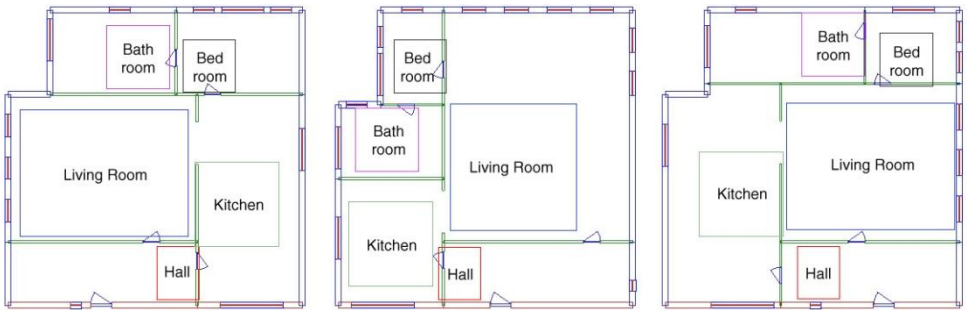


Figure 4: By placing some feature areas we can lay out the positions of rooms inside a building.

We use features in the form of a path to position road-side objects. With our basic constraint possibilities, we can define rules on the distance between the objects, and by adding a rule that they should be located on the path features next to the road, we can easily add objects on roads with different widths and properties. In our example in Figure 5, we created two roads with different widths and lanes on the same path. In the top example, no street lights are added, and therefore delineators with a reflector are used as an alternative, which is common for smaller roads in a rural area. The traffic sign that denotes the speed limit is placed at the border of a zone marked with a feature area.

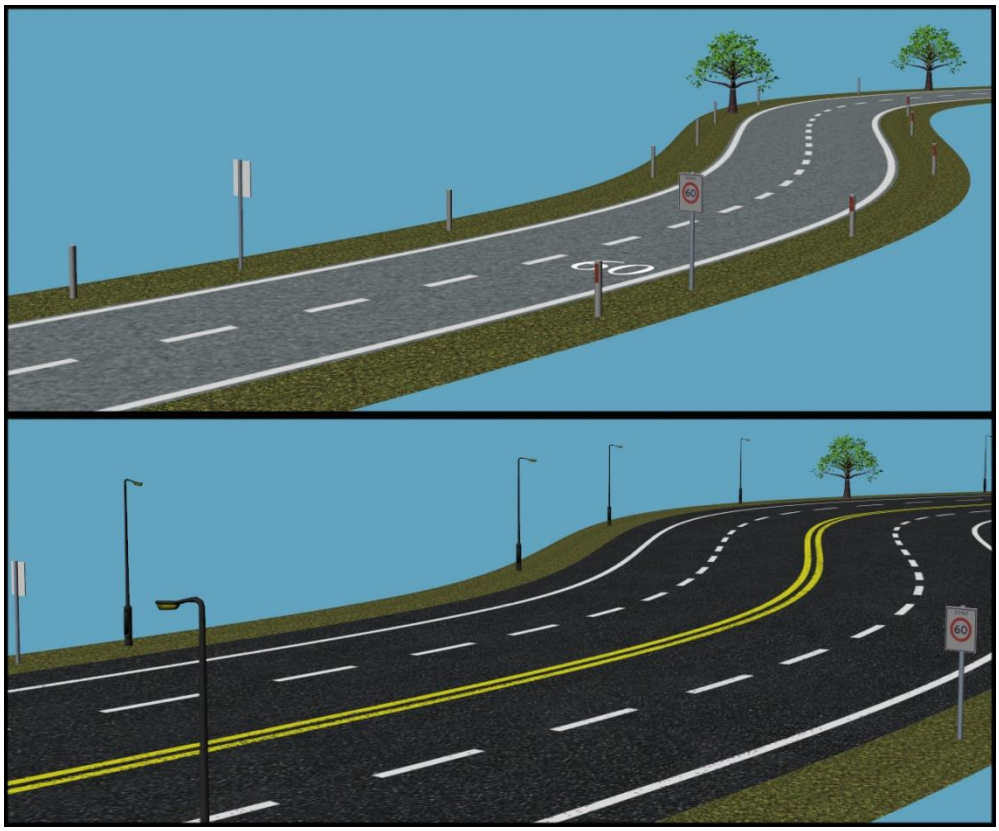


Figure 5: The road side objects are automatically adjusted to a different road width and when no street lights are placed, delineators with a reflector are added (see top image). The top scene contains 49 objects and took 620ms; the bottom scene contains 32 objects and took 318ms.

These examples show that our solving approach is generic and applicable to many different layout problems common to procedural generation of game worlds. Throughout the paper, we focused on creating room interiors, since it offers many different examples of layout problems that can be handled by our solving approach. But the same approach, only with other classes, each with their proper placement rules, can be applied to floor plan creation and road-side scenarios. This is due to the fact that we allow for abstract feature-based representations of class instances.

6 CONCLUSION

In the ever increasing game worlds, it is becoming impossible for a designer to create every part of the game world by hand. We proposed and implemented a layout solving approach that presents a solution to speed up manual design methods, by solving some placement rules of newly placed objects, and that can also be used to create parts of the game world automatically.

We highlighted that our approach is capable of generating room interiors fast enough to be usable at runtime, i.e. by creating interior layouts only when necessary. This way, a huge game world in an urban environment could be extended with the possibility of entering buildings without designers spending weeks and months on creating every

layout by hand. They do however remain in control over the generation procedure, by creating the plans that decide which objects are added to the layout, and by defining the rules that are to be followed when placing these objects. This also means that the knowledge we have about how best to plan a layout, and how objects relate to one another, can be captured in the plan and the class rules.

To keep the approach generic, we chose the use of abstract class representations with the help of features to allow the class rules to be reused for different objects belonging to the same class. Moreover this allows the system to be used in many different scenarios. New classes can easily be created, just by defining a representation and some general rules that should apply to all instances of that class. We showed some different interior layout examples, as well as a floor plan layout and a scene with some road-side objects, all created with the same solving mechanism. We can conclude that our approach is usable in both automatic and user-assisted design, and that it can solve different kinds of layout problems that are commonly noticed in game world creation.

We are currently integrating our solving approach with a more comprehensive semantics-based class library that stores the knowledge base for the used objects (Tutenel, et al. (2009)) in order to capture even more user knowledge in the class definitions and in the rules as well. We also plan to expand on the system we created based on our approach with a clear and visual editor for the classes, the rules and the plans.

REFERENCES

Assassin's Creed (2007). *http://assassinscreed.uk.ubi.com/experience/* Retrieved March 10, 2009

Calderon, C., Cavazza, M., and Diaz, D. (2003). A New Approach to the Interactive Resolution of Configuration Problems in Virtual Environments. In *Proceedings of Smart Graphics 2003*, Heidelberg, Germany.

Charman, P. (1993). Solving Space Planning Problems Using Constraint Technology. Institute of Cybernetics - Estonian Academy of Sciences.

Coyne, B., and Sproat, R. (2001). WordsEye: an Automatic Text-to-Scene Conversion System. In *Proceedings of International Conference on Computer Graphics and Interactive Technologies (SIGGRAPH 2001)*, Los Angeles, California, USA.

Gaildrat, V. (2007). Declarative Modelling of Virtual Environments, Overview of issues and Applications. In *International Conference on Computer Graphics and Artificial Intelligence (3IA 2007)*, Athens, Greece.

Grand Theft Auto IV (2008). *http://www.rockstargames.com/IV/* Retrieved March 10, 2009

Hahn, E., Bose, P., and Whitehead, A. (2006). Persistent Realtime Building Interior Generation. In *Proceedings of the 2006 ACM SIGGRAPH Symposium on Videogames*, Boston, Massachusetts, USA.

Honda, K., and Mizoguchi, F. (1995). Constraint-Based Approach for Automatic Spatial Layout Planning. In *Proceedings of the 11th Conference on Artificial Intelligence for Applications, 1995*, Los Angeles, California, USA.

Larive, M., and Gaildrat, V. (2006). Wall Grammar for Building Generation. In *Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia*, Kuala Lumpur, Malaysia.

Le Roux, O., Gaildrat, V., and Caubet, R. (2001). Using Constraint Propagation and Domain Reduction for the Generation Phase in Declarative Modeling. In *Conference on Information Visualisation, IV'2001*, London, UK.

Le Roux, O., Gaildrat, V., and Caubet, R. (2004). Constraint Satisfaction Techniques for the Generation Phase in Declarative Modeling. In M. Sarfraz (Ed.). *Geometric Modeling: Techniques, Applications, Systems and Tools* (pp. 194-215): Kluwer Academic Publishers Norwell, MA, USA.

Martin, J. (2006). Procedural House Generation: A Method for Dynamically Generating Floor Plans. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, Redwood City, California, USA.

Müller, P., Vereenooghe, T., Wonka, P., Paap, I., and Van Gool, L. (2006). Procedural 3D Reconstruction of Puuc Buildings in Xkipché. In *Proceedings 7th International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage - VAST2006*, Nicosia, Cyprus.

Müller, P., Wonka, P., Haegler, S., Ulmer, A., and Van Gool, L. (2006). Procedural Modeling of Buildings. *ACM Transactions on Graphics, 25*(3), 614-623.

Pfefferkorn, C. E. (1975). A Heuristic Problem Solving Design System for Equipment or Furniture Layouts. *Communications of the ACM, 18*(5), 286-297.

Rau-Chaplin, A., MacKay-Lyons, B., and Spierenburg, P. (1996). The LaHave House Project: Towards an Automated Architectural Design Service. In *Proceedings of the International Conference on Computer-Aided Design (CADEX'96)*, Hagenberg, Austria.

Rau-Chaplin, A., and Smedley, T. J. (1997). A Graphical Language for Generating Architectural Forms. In *Proceedings of the 1997 IEEE Symposium on Visual Languages*, Capri, Italy.

Smelik, R. M., de Kraker, K. J., Groenewegen, S. A., Tutenel, T., and Bidarra, R. (2009). A Survey of Procedural Methods for Terrain Modelling. *TBP in Proceedings of CASA Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS)*, Amsterdam, The Netherlands.

Smith, G., Salzman, T., and Stuerzlinger, W. (2001). 3D Scene Manipulation with 2D Devices and Constraints. In *Graphics Interface Proceedings 2001*, Ottawa, Ontario, Canada.

Tutenel, T., Bidarra, R., Smelik, R. M., and de Kraker, K. J. (2008). The Role of Semantics in Games and Simulations. *Computers in Entertainment, 6*(4), 1-35.

Tutenel, T., Bidarra, R., Smelik, R. M., and de Kraker, K. J. (2009). Using Semantics to Improve the Design of Game Worlds. (Submitted for publication).

Wonka, P., Wimmer, M., Sillion, F., and Ribarsky, W. (2003). Instant Architecture. *ACM Transactions on Graphics (TOG), 22*(3), 669-677.

Xu, K., Stewart, J., and Fiume, E. (2002). Constraint-Based Automatic Placement for Scene Composition. In *Graphics Interface Proceedings 2002*, University of Calgary.

Yong, L., Congfu, X. U., Zhigeng, P., and Yunhe, P. (2004). Semantic Modeling Project: Building Vernacular House of Southeast China. In *Proceedings of the 2004 ACM SIGGRAPH International Conference on Virtual Reality Continuum and its Applications in Industry*, Singapore.

A Survey of Procedural Methods for Terrain Modelling*

Ruben M. Smelik, Klaas Jan de Kraker and Saskia A. Groenewegen
TNO Defence, Security and Safety
The Hague, The Netherlands

Tim Tutenel and Rafael Bidarra
Delft University of Technology
Delft, The Netherlands

Abstract

Procedural methods are a promising but underused alternative to manual content creation. Commonly heard drawbacks are the randomness of and the lack of control over the output and the absence of integrated solutions, although more recent publications increasingly address these issues. This paper surveys procedural methods applied to terrain modelling, evaluating realism of their output, performance and control users can exert over the procedure.

1 Introduction

In the last two decades, 3D virtual worlds have advanced from primitive to, at least visually, very advanced and complex. However, the modelling process, tools, and techniques used to create these worlds have not advanced that much: they are still laborious and repetitious in use and require specialized 3D modelling skills.

Procedural modelling has been an active research topic for at least thirty years. The philosophy of procedural modelling is, instead of designing content by hand, to design a procedure that creates content automatically. This approach has been successfully applied to generate, for example, textures, geometric models, animations and even sound clips. A major topic within procedural modelling is the automatic generation of terrain models, which started with natural phenomena such as terrain elevation and growth of plants in the 1980s and 1990s and extended its focus to urban environments at the start of the new millennium.

Despite the encouraging results, procedural modelling is not often applied in mainstream terrain modelling. Several factors limit this transition from manual to automated modelling. For one, both research papers and commercial tools typically focus on one aspect of terrain modelling (for instance, generating interesting elevation profiles) and address other aspects to a limited extent or not at all. The integration and adjustment of existing procedural methods in such a way that they can automatically generate a complete and consistent terrain model remains to date largely unsolved. Another known issue of procedural methods is the lack of control they offer to users. The inherent randomness of the resulting content often forces users to model by trial and error. More recent publications sometimes address this issue with specific solutions, but overall it has not yet received sufficient attention.

This paper presents a short survey of procedural methods applied to terrain modelling. We discuss important properties of the methods, such as the realism of the output, the performance of the algorithm and the facilities it provides users to influence and control the generation process. The purpose of this paper is twofold: to give readers interested in the field of procedural terrain modelling an overview of the research to date, and to assess the extent to which the identified issues of procedural methods are being addressed and what remains to be done.

In (Smelik et al., 2008, 2009), we have identified several requirements a procedural modelling framework should fulfil to be a qualitatively acceptable and more productive alternative to the

*This research has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie).

current modelling workflow. We described the conceptual design of such a framework. It integrates procedural methods and manages dependencies between terrain features in order to generate a complete, consistent terrain model, which matches a sketch of the rough layout of the terrain made by the user. It involved distinguishing several layers in a terrain model, each containing natural features (earth, water, vegetation layers) and man-made (road and urban layers). We present the design, implementation and results of two of these layers. In this paper, the distinction of terrain layers is also practical to structure the work we surveyed.

The next sections discuss procedural methods for elevation data, water bodies, vegetation, road networks and urban environments, followed by a conclusion on the state of the art.

2 Height-maps

Height-maps, i.e. two-dimensional grids of elevation values, are often used as the basis of a terrain model. There are many procedural algorithms for creating height-maps.

Among the oldest algorithms are the subdivision based methods. A coarse height-map is iteratively subdived, each iteration using controlled randomness to add detail. Miller (1986) describes several variants of the well known mid-point displacement method, in which a new point's elevation is set to the average of its corners in a triangle or diamond shape plus a random offset. The offset's range decreases each iteration according to a parameter that controls the roughness of the resulting height-map.

Height-map generation is nowadays often based on fractal noise generators (Fournier et al., 1982; Voss, 1985), such as Perlin noise (Perlin (1985)), which generates noise by sampling and interpolating points in a grid of random vectors. Rescaling and adding several levels of noise to each point in the height-map results in natural, mountainous-like structures. For a recommended textbook on fractal noise and height-map generation, see Ebert et al. (2003).

These height-maps can be transformed further based on common imaging filters (e.g. smoothing) or on simulations of physical phenomena, for instance erosion. Thermal erosion diminishes sharp changes in elevation, by iteratively distributing material from higher to lower points, until the talus angle, i.e. maximum angle of stability for a material such as rock or sand, is reached. Erosion caused by rainfall (fluvial erosion) can be simulated using, for example, cellular automata, where the amount of water and dissolved material that flows out to other cells is calculated based on the local slope of the terrain surface. Musgrave treats both types of erosion (Musgrave et al., 1989; Musgrave, 1993) and Olsen (2004) discusses several speed optimizations with reduced but acceptable quality. Beneš and Forsbach (2001) introduce a terrain structure suited for more realistic erosion algorithms. Their terrain model consists of stacked horizontal slices of material, each having an elevation value and material properties, e.g. density. It is a trade-off between the limited but efficient height-map structure and a full voxel terrain. The model also allows for air layers, thereby it supports cave structures.

While these erosion algorithms add much to the believability of mountainous terrain, they are also notoriously slow, having to run for hundreds to thousands of iterations. Recent research has focussed on interactive erosion algorithms, often by porting algorithms to the GPU. Promising examples include (Anh et al., 2007) and (Šťava et al., 2008).

The basic noise-based height-map generation delivers results that are fairly random; users control the outcome only on a global level, often using unintuitive parameters. Several researchers have addressed this issue. Stachniak and Stuerzlinger (2005) propose a method that integrates constraints (expressed as mask images) into the terrain generation process. It employs a search algorithm that finds an acceptable set of deformation operations to apply to a random terrain in order to obtain a terrain that conforms to these constraints. Schneider et al. (2006) introduce an editing environment in which the user edits the terrain by interactively modifying the base functions of the noise generator (by replacing the Perlin noise grid with a set of user-drawn gray-scale images). Zhou et al. (2007) describe a technique that generates terrain based on an example input height-map and a user line drawing that defines the occurrence of large-scale curved line features, such as a mountain ridge. Features are extracted from the example height-map and matched to the sketched curves and seamed in the resulting height-map. De Carpentier and

Bidarra (2009) introduce procedural brushes: users paint height-mapped terrain directly in 3D by applying simple terrain raising brushes but also GPU-based brushes that generate several types of noise in real-time (see Fig. 1a). Saunders (2006) proposes a very different method, which synthesizes a height-map based on Digital Elevation Models (DEM) of real world terrain. A user of his system Terrainosaurus draws a 2D map of polygonal regions, each of which is marked to have a certain elevation profile. For realism, the straight boundaries of the region are perturbed and then rasterized in a grid. A height-map is instantiated using a genetic algorithm that selects DEM data that matches the requested elevation profile. Kamal and Uddin (2007) present a constrained mid-point displacement algorithm that creates a single mountain according to such properties as elevation and base spread. Belhadj (2007) introduces a more general system where a set of known elevation values constrain the mid-point displacement process. Possible applications are interpolation of coarse or incomplete DEM's or user line sketches.

An inherit limitation of height-maps is that they do not support rock overhangs and caves. Gamito and Musgrave (2001) propose a terrain warping system that results in regular, artificial overhangs. A recent method (Peytavie et al., 2009) provides a more elaborate structure with different material layers that supports rocks, arches, overhangs and caves. Their resulting terrain models are visually very plausible and natural.

As an illustration of the state of the art in tool support, see Fig. 1d) for a render of a height-map generated by L3DT (Torpy, 2009), one of the commercial tools for creating height-maps.

3 Rivers, Oceans and Lakes

For generating rivers, several authors have proposed algorithms that run either during or after height-map generation. Kelley et al. (1988) take a river network as the basis of a height-map. They start with a single straight river and recursively subdivide it, resulting in a stream network. This forms a skeleton for the height-map, which is then filled using a scattered data interpolation function. The climate type and the soil material influence the shape of the stream network.

Prusinkiewicz and Hammel (1993) combine the generation of a curved river with a height-map subdivision scheme. Of the river's starting triangle, one edge is marked as the entry and one as the exit of the river. In a subdivision step, the triangle is divided into smaller triangles, and the river's course from entry to exit can now take several alternative forms. The elevation of the triangles containing the river is set to be the sum of the negative displacements of the river on all recursion levels (resulting in a river bed), other triangles are processed using standard mid-point displacement. After eight or more recursions, the resulting river course looks reasonably natural. A major downside of the approach is that the river is placed at a constant and low elevation level, and thus carves unnaturally deep through a mountainous terrain.

A more advanced approach described by Belhadj and Audibert (2005) creates a height-map with mountain ridges and river networks. Starting with an empty map, they place pairs of ridge particles at a particular high elevation and move them in opposite directions in several iterations. A Gaussian curve is drawn on the height-map along the particle positions of each iteration. Next, they place river particles along the top of the mountain ridge and let them flow downwards according to simple physics (comparable to hydraulic erosion). The remaining points in between ridges and rivers are filled with an inverse midpoint displacement technique. For this specific type of terrain, i.e. steep mountain ridges with valleys featuring a dense river network, the method is fast and effective.

Except for rivers, procedural water bodies, such as oceans and lakes and their connections, stream networks, deltas and waterfalls, have received too little attention to date. The forming of lakes is typically not considered at all. Oceans are commonly generated setting a fixed water level (e.g. 0 m) or by starting a flooding algorithm from points of low elevation. Teoh (2008) also states that the research in this area is incomplete: several river and coastal features have not been addressed. He proposes fast and simple algorithms for river meandering, deltas and beach forming, which require further work to increase their realism.

4 Plant Models and Vegetation Distribution

Regarding vegetation, authors developed several procedures for generating tree and plant models and methods for automatic placement of vegetation on a terrain model. The former can be used to quickly obtain a set of similar but varying plant models of the same species; the latter saves terrain modellers the laborious task of manually placing all these individual vegetation models in a large forest.

Procedural plant models grow, starting from the root, adding increasingly smaller branches and ending with the leaves. They are based on grammar rewriting. Prusinkiewicz and Lindenmayer (1990) discuss the Lindenmayer-system, or *L-system*, an often used rewriting system. They explain how production rules can be applied in 3D, and present many examples of generated trees together with their grammar.

Lintermann and Deussen (1999) propose a more intuitive system to procedurally model plants, by placing plant components (e.g. a leaf) in a graph. Connected components can be structured in subgraphs (e.g. a twig). The system traverses this graph, generating and placing instances of the components in an intermediate graph that is used for geometry generation. Fig. 1e) shows a tree created with their commercial plant modelling software XFrog.

Deussen et al. (1998) describe an ecosystem simulation model to populate an area with vegetation. The input of this simulation model is the height-map and a water map, several ecological properties of plant species, such as rate of growth, and, optionally, an initial distribution of plants. Based on this and taking into account rules for competition for soil, sunlight and water, a distribution of plants inside an area is iteratively determined (see Fig. 1b), running for several minutes.

Another procedure for vegetation placement by Hammes (2001) is based on ecosystems. He uses elevation data, relative elevation, slope, slope direction and multi fractal noise to select one of the defined ecosystems. Ground vegetation textures are generated at run-time, depending on the level of detail and the ecosystem. The ecosystem also determines the number of plants per species, which are then placed randomly.

Procedural modelling of vegetation delivers believable results and is already applied quite often in modern games, for instance using the commercial package SpeedTree.

5 Road Networks

The generation of road networks for cities can be done using a variety of methods, of which we treat the pattern-based approaches, L-systems, agent simulations and tensor fields. The simplest technique is to generate a dense square grid (as in Greuter et al. (2003)). Displacement noise can be added to grid points to create a less repetitive network, but still the realism of this technique is limited.

A more elaborate method to create roads is via the use of templates, as proposed by Sun et al. (2002). They observe several frequent patterns in real road networks and aim to reconstruct them. For each pattern, there is a corresponding template: a population-based template (implemented as the Voronoi diagram of population centres), a raster and radial template, or a mixed template. The main arteries of the road map are the highways, which are generated first using these pattern templates. Simple rules are applied to check their validity. When encountering impassable areas (e.g. oceans), they are discarded or diverted. Next, the main roads are often curved to avoid large elevation gradients. The regions they encompass are filled in with a raster of streets.

Parish and Müller (2001) use an extended L-system to grow their road network. The L-system is goal-driven and the goals are the population density (the roads try to connect population centres) and specific road patterns. Examples of such patterns are the raster or the radial pattern. Their L-system is extended with rules that have a tendency to connect new proposed roads to existing intersections and rules that check road validity with respect to impassable terrain and elevation constraints. Streets are also inserted into the remaining areas as simple grids.

Kelly and McCabe (2007) introduce the interactive city editor CityGen, in which a user defines the main roads by placing nodes in the 3D terrain. Regions enclosed by these roads can be filled with one of three patterns: Manhattan-style grids, industrial grown roads with dead-ends and organic roads as in e.g. North-American suburbs.

Glass et al. (2006) describe several experiments of replicating the road structure found in South African informal settlements using a combination of a Voronoi diagram for the major roads with L-systems or regular subdivision with and without displacement noise for the minor roads. They were reasonably successful in recreating the observed patterns.

Different from the grammar- and pattern-based approaches discussed above, Lechner et al. (2003) take an agent-based approach, in which they divide the city into areas including not only residential, commercial and industrial areas, but also special areas like government buildings, squares, and institutions. They place two agents, named the *extender* and the *connector*, at a seed position in the terrain map. The extender searches for unconnected areas in the city. When it finds such an area that is located not too far from the existing road network, it finds the most suitable path to connect the area to the network. The connector agent starts from a certain location on the existing network and randomly chooses another spot on the network, within a certain radius. It checks the length of the shortest path to this spot. If the travel time is considered too long, a direct road connection is added to the network. In Lechner et al. (2006), the authors extend this method with, among other things, agents that are responsible for constructing main roads for fast connections through the city, and agents that develop small streets. This method gives plausible results, but a disadvantage is its very long running time.

Chen et al. (2008) propose interactive modelling of road networks by the use of tensor fields. They define how to create common road patterns (grid, radial, along a boundary) using tensor fields. A road network is generated from a tensor field, by tracing the streamlines from seed points in the major eigenvector direction until a stopping condition is met. Next, along this traced curve new seed points are placed for tracing streamlines in the perpendicular (minor eigenvector) direction. Users can place new basis tensor fields, such as a radial pattern, smooth the field, or use a brush to locally constrain the field in a specific direction. Noise can be applied to make the road network less regular and thereby more plausible.

In the discussed methods, the influence of the underlying terrain map and elevation profile is to varying degrees taken into account. Most methods take only basic measures to avoid too steep ascending roads and roads through water bodies. Kelly and McCabe (2007) plan the precise path of their main roads between the user set nodes to have an even change in elevation as much as possible. Still, for rough terrain this measure will not be adequate and the terrain needs to be modified to accommodate for the road. Bruneton and Neyret (2008) propose a simple and effective method for blending road profiles into the height-map using shaders.

6 Urban Environments

Kelly and McCabe (2006) give an elaborate overview of several approaches for generating urban environments. Watson et al. (2008) give a practical overview of the state of the art.

The common approach for procedurally generating cities is to start from a dense road network and identify the polygonal regions enclosed by streets. Subdivision of these regions results in lots, for which different subdivision methods exist, see e.g. Parish and Müller (2001) or Kelly and McCabe (2007). To populate these lots with buildings, either the lot shape is used directly as the footprint of a building, or a building footprint is fitted on the lot. By simply extruding the footprint to a random height, one can generate a city of skyscrapers and office buildings. To obtain more interesting building shapes, several approaches have been devised.

Greuter et al. (2003) generate office buildings by combining several primitive shapes into a floor plan and extruding these to different heights. Parish and Müller (2001) start with a rectangular floor plan and apply an L-system to refine the building. Both approaches are most useful for relatively simple office building models.

Wonka et al. (2003) introduce the concept of the *split grammar*, a formal context-free grammar designed to produce building models. The split grammar resembles an L-system, but is based on shapes as primitive elements rather than symbols. In their system, a specific building style can be acquired by setting an attribute of the start symbol, which is propagated during the rewrite. Within one building model, the style can differ per floor (e.g. an apartment building with shops on the ground floor). Their approach focuses mostly on generating coherent and believable facades

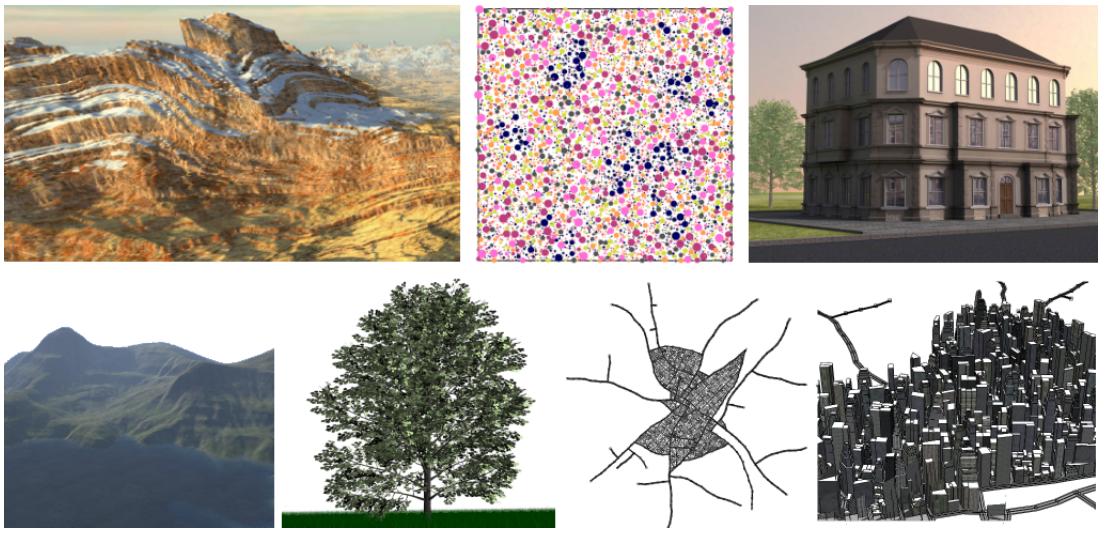


Figure 1: a) Terrain interactively created using procedural brushes (De Carpentier and Bidarra, 2009). b) Plant distribution simulation (Deussen et al., 1998). c) Complex building facade (Finkenzeller and Bender, 2008). d) A 3D render of a height-map generated using L3DT (Torpy, 2009). e) A plant model generated using XFrog (Greenworks, 2009). f) and g) A road network and corresponding city generated using CityEngine (Procedural, inc., 2009).

for relatively simple shaped buildings. Larive and Gaildrat (2006) use a similar kind of grammar, called a *wall grammar*. With this grammar they are able to generate building walls with additional geometric details, such as balconies.

Müller et al. (2006) apply another type of grammar, named *shape grammar*. The main property of a shape grammar is that it uses context-sensitive rules, whereas a split grammar uses context-free rules, which in this case allows the possibility to model roofs and rotated shapes. They start with a union of several volumetric shapes which defines the boundary of the building. This shape is then divided into floors and the resulting facades are subdivided into walls, windows, and doors by means of a grammar system. In a final step, the roof is constructed on top of the building. Fig. 1f shows a road network and Fig. 1g the corresponding city generated by their commercial product, CityEngine (Procedural, inc., 2009). Besides the well known business buildings, the grammar can also model residential buildings, e.g. suburban homes or ancient Roman villas.

Although the shape grammars in Müller et al. (2006) can generate visually convincing building models, Finkenzeller and Bender (2008) note that semantic information regarding the role of each shape within the complete building is missing. They propose to capture this semantic information in a typed graph. Their workflow consists of three steps. Starting with a rough building outline, a building style graph can be applied to this model. This results in an intermediate semantic graph representation of the building, which can be modified or regenerated with a different style. In the last step, geometry is created based on the intermediate model, and textures are applied, resulting in a complete 3D building. Finkenzeller (2008) describes in more detail the generation of facades and roofs in their system (see Fig. 1c)).

Yong et al. (2004) describe a method to create vernacular-style southeast Chinese houses using an extended shape grammar. The grammar is hierarchical and starts at the city level (whereas in other methods a shape grammar is applied to an individual building footprint). The grammar then produces streets, housing blocks, roads, and in further productions houses with components such as gates, windows, walls, and roofs. Through a number of control rules (defining, for instance, component ratio constraints) the validity of the buildings can be asserted. By applying this grammar system, a typical ancient southeast Chinese town can be generated with plausible results, since the building style of these towns is very rigidly structured.

Müller et al. (2007) used a very different approach for constructing building facades. Their method takes a single image of a facade of a real building as input and is able to reconstruct a detailed 3D facade model, using a combination of imaging and shape grammar generation.

Although the above methods give fast and visually pleasing results, the cities they generate often lack a realistic structure. New research incorporates existing urban land use theories and models in the generation process. Groenewegen et al. (2009) present a method that generates a distribution of different types of districts according to land use models of cities in Western-Europe and North-America. It takes into account a large number of relevant factors, including the historic core of the city and the attraction certain types of terrain (hillsides, oceans, rivers) have for e.g. industrial or high-class residential districts. Weber et al. (2009) use comparable (albeit slightly simplified) models for a simulation of expanding cities over time. Their system is fast (about 1 sec. per simulated year) and interactive, meaning that the user can guide the simulation by changing roads or painting land use values on the terrain.

7 Conclusions

Procedural methods for terrain modelling are becoming increasingly attractive for both academia and industry, as a promising alternative to the expensive manual creation of content for virtual worlds. We have classified these methods into five main areas, and discussed a large variety of research approaches and results of each of them: terrain elevation, water elements, vegetation, road networks and urban environments.

From its early years, where the focus was mainly set on height-map generation, until now, with a shift towards more and more realistic urban environments, there is a considerable body of research results available. Many basic procedural methods deploy common building blocks such as noise, rewriting grammars and simple simulation systems, of which a large number of variants, often very much domain-specific, are now being proposed, in particular within the road and urban categories. Of all categories discussed, the water-related area is clearly the most underdeveloped.

Regarding upcoming research, several interesting trends have been identified. Among them, three promising directions can be summarized as follows. First, performance and interactivity of procedural methods will continue to improve, often by means of parallel programming on the GPU. Second, road networks and urban areas will certainly continue to improve in variation and level of detail, but the realism leap will likely be given by deploying more and more semantics in both the procedural generation process and the generated models (Tutenel et al., 2008). And last, the key to a widespread deployment of procedural methods by non-experts (e.g. game designers, artists, scenario designers) will be the integration of procedural methods within a framework, offering among other things, more intuitive controls, tools to generate complete terrain models and non-intrusive mechanisms to maintain the consistency among generated features (Smelik et al., 2009).

References

Anh, N. H., Sourin, A., and Aswani, P. (2007). Physically based Hydraulic Erosion Simulation on Graphics Processing Unit. In *GRAPHITE '07: Proceedings of the* 5^{th} *International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*, pages 257–264, New York, NY, USA. ACM.

Belhadj, F. (2007). Terrain Modeling: a Constrained Fractal Model. In Spencer, S. N., editor, *AFRIGRAPH '07: Proceedings of the* 5^{th} *International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, pages 197–204, New York, NY, USA. ACM.

Belhadj, F. and Audibert, P. (2005). Modeling Landscapes with Ridges and Rivers: Bottom Up Approach. In *GRAPHITE '05: Proceedings of the* 3^{rd} *International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, pages 447–450, New York, NY, USA. ACM.

Beneš, B. and Forsbach, R. (2001). Layered Data Representation for Visual Simulation of Terrain Erosion. In *SCCG '01: Proceedings of the* 17^{th} *Spring Conference on Computer Graphics*, pages 80–86, Washington, DC, USA. IEEE Computer Society.

Bruneton, E. and Neyret, F. (2008). Real-time Rendering and Editing of Vector-based Terrains. In Drettakis, G. and Scopigno, R., editors, *Eurographics 2008 Proceedings*, volume 27, pages 311–320, Hersonissos, Grèce.

Chen, G., Esch, G., Wonka, P., Müller, P., and Zhang, E. (2008). Interactive Procedural Street Modeling. In *SIGGRAPH '08: Proceedings of the* 35^{th} *Annual Conference on Computer Graphics and Interactive Techniques*, volume 27, pages 1–10, New York, NY, USA. ACM.

De Carpentier, G. and Bidarra, R. (2009). Interactive GPU-based Procedural Heightfield Brushes. In *Proceedings of the* 4^{th} *International Conference on the Foundation of Digital Games*, Florida, USA.

Deussen, O., Hanrahan, P., Lintermann, B., Měch, R., Pharr, M., and Prusinkiewicz, P. (1998). Realistic Modeling and Rendering of Plant Ecosystems. In *SIGGRAPH '98: Proceedings of the* 25^{th} *Annual Conference on Computer Graphics and Interactive Techniques*, pages 275–286, New York, NY, USA. ACM.

Ebert, D. S., Worley, S., Musgrave, F. K., Peachey, D., and Perlin, K. (2003). *Texturing & Modeling, a Procedural Approach*. Elsevier, 3^{rd} edition.

Finkenzeller, D. (2008). Detailed Building Facades. *IEEE Computer Graphics and Applications*, 28(3):58–66.

Finkenzeller, D. and Bender, J. (2008). Semantic Representation of Complex Building Structures. In *Computer Graphics and Visualization (CGV 2008) - IADIS Multi Conference on Computer Science and Information Systems*, Amsterdam, The Netherlands.

Fournier, A., Fussell, D., and Carpenter, L. (1982). Computer Rendering of Stochastic Models. *Communications of the ACM*, 25(6):371–384.

Gamito, M. and Musgrave, F. K. (2001). Procedural Landscapes with Overhangs. In 10^{th} *Portuguese Computer Graphics Meeting*, pages 33–42.

Glass, K. R., Morkel, C., and Bangay, S. D. (2006). Duplicating Road Patterns in South African Informal Settlements Using Procedural Techniques. In Spencer, S. N., editor, *AFRIGRAPH '06: Proceedings of the* 4^{th} *International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, pages 161–169, New York, NY, USA. ACM.

Greenworks (2009). XFrog. Available from http://www.xfrogdownloads.com.

Greuter, S., Parker, J., Stewart, N., and Leach, G. (2003). Real-time Procedural Generation of 'Pseudo Infinite' Cities. In *GRAPHITE '03: Proceedings of the* 1^{st} *International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, pages 87–94, New York, NY, USA. ACM.

Groenewegen, S. A., Smelik, R. M., de Kraker, K. J., and Bidarra, R. (2009). Procedural City Layout Generation Based On Urban Land Use Models. In Alliez, P. and Magnor, M., editors, *Short Paper Proceedings of Eurographics 2009*, pages 45–48, Munich, Germany. Eurographics Association.

Hammes, J. (2001). Modeling of Ecosystems as a Data Source for Real-Time Terrain Rendering. In *DEM '01: Proceedings of the First International Symposium on Digital Earth Moving*, pages 98–111, London, UK. Springer-Verlag.

Kamal, K. R. and Uddin, Y. S. (2007). Parametrically Controlled Terrain Generation. In *GRAPHITE '07: Proceedings of the* 5^{th} *International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*, pages 17–23, New York, NY, USA. ACM.

Kelley, A. D., Malin, M. C., and Nielson, G. M. (1988). Terrain Simulation Using a Model of Stream Erosion. In *SIGGRAPH '88: Proceedings of the* 15^{th} *Annual Conference on Computer Graphics and Interactive Techniques*, pages 263–268, New York, NY, USA. ACM.

Kelly, G. and McCabe, H. (2006). A Survey of Procedural Techniques for City Generation. *Institute of Technology Blanchardstown Journal*, 14:87–130.

Kelly, G. and McCabe, H. (2007). Citygen: An Interactive System for Procedural City Generation. In *Proceedings of GDTW 2007: The Fifth Annual International Conference in Computer Game Design and Technology*, pages 8–16, Liverpool, UK.

Larive, M. and Gaildrat, V. (2006). Wall Grammar for Building Generation. In *GRAPHITE '06: Proceedings of the* 4^{th} *International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia*, pages 429–437, New York, NY, USA. ACM.

Lechner, T., Ren, P., Watson, B., Brozefski, C., and Wilenski, U. (2006). Procedural Modeling of Urban Land Use. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Research posters*, page 135, New York, NY, USA. ACM.

Lechner, T., Watson, B., Wilensky, U., and Felsen, M. (2003). Procedural City Modeling. In 1^{st} *Midwestern Graphics Conference*, St. Louis, MO, USA.

Lintermann, B. and Deussen, O. (1999). Interactive Modeling of Plants. *IEEE Computer Graphics and Applications*, 19(1):56–65.

Miller, G. S. P. (1986). The Definition and Rendering of Terrain Maps. In *SIGGRAPH '86: Proceedings of the* 13^{th} *Annual Conference on Computer Graphics and Interactive Techniques*, volume 20, pages 39–48, New York, NY, USA. ACM.

Müller, P., Wonka, P., Haegler, S., Ulmer, A., and Gool, L. V. (2006). Procedural Modeling of Buildings. In *SIGGRAPH '06: Proceedings of the* 33^{rd} *Annual Conference on Computer Graphics and Interactive Techniques*, pages 614–623, New York, NY, USA. ACM.

Müller, P., Zeng, G., Wonka, P., and Gool, L. V. (2007). Image-based Procedural Modeling of Facades. In *SIGGRAPH '07: Proceedings of the* 34^{th} *Annual Conference on Computer Graphics and Interactive Techniques*, volume 26, New York, NY, USA. ACM.

Musgrave, F. K. (1993). *Methods for Realistic Landscape Imaging*. PhD thesis, Yale University, New Haven, CT, USA.

Musgrave, F. K., Kolb, C. E., and Mace, R. S. (1989). The Synthesis and Rendering of Eroded Fractal Terrains. In *SIGGRAPH '89: Proceedings of the* 16^{th} *Annual Conference on Computer Graphics and Interactive Techniques*, pages 41–50, New York, NY, USA. ACM.

Olsen, J. (2004). Realtime Procedural Terrain Generation. Technical Report, University of Southern Denmark.

Parish, Y. I. H. and Müller, P. (2001). Procedural Modeling of Cities. In *SIGGRAPH '01: Proceedings of the* 28^{th} *Annual Conference on Computer Graphics and Interactive Techniques*, pages 301–308, New York, NY, USA. ACM.

Perlin, K. (1985). An Image Synthesizer. In *SIGGRAPH '85: Proceedings of the* 12^{st} *Annual Conference on Computer Graphics and Interactive Techniques*, volume 19, pages 287–296. ACM.

Peytavie, A., Galin, E., Merillou, S., and Grosjean, J. (2009). Arches: a Framework for Modeling Complex Terrains. In *Eurographics 2009 Proceedings*. Eurographics Association.

Procedural, inc. (2009). CityEngine. Available from http://www.procedural.com.

Prusinkiewicz, P. and Hammel, M. (1993). A Fractal Model of Mountains with Rivers. In *Proceeding of Graphics Interface '93*, pages 174–180.

Prusinkiewicz, P. and Lindenmayer, A. (1990). *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, NY, USA.

Saunders, R. L. (2006). Terrainosaurus: Realistic Terrain Synthesis Using Genetic Algorithms. Master's thesis, Texas A&M University.

Schneider, J., Boldte, T., and Westermann, R. (2006). Real-Time Editing, Synthesis, and Rendering of Infinite Landscapes on GPUs. In *Vision, Modeling and Visualization 2006*.

Smelik, R., de Kraker, K. J., Tutenel, T., and Bidarra, R. (2009). Declarative Terrain Modeling for Military Training Games. *Submitted for publication*.

Smelik, R., Tutenel, T., de Kraker, K. J., and Bidarra, R. (2008). A Proposal for a Procedural Terrain Modelling Framework. In *Poster Proceedings of the* 14^{th} *Eurographics Symposium on Virtual Environments EGVE08*, pages 39–42, Eindhoven, The Netherlands.

Stachniak, S. and Stuerzlinger, W. (2005). An Algorithm for Automated Fractal Terrain Deformation. *Computer Graphics and Artificial Intelligence*, 1:64–76.

Sun, J., Yu, X., Baciu, G., and Green, M. (2002). Template-based Generation of Road Networks for Virtual City Modeling. In *VRST '02: Proceedings of the ACM symposium on Virtual Reality Software and Technology*, pages 33–40, New York, NY, USA.

Teoh, S. T. (2008). River and Coastal Action in Automatic Terrain Generation. In Arabnia, H. R. and Deligiannidis, L., editors, *CGVR 2008: Proceedings of the 2008 International Conference on Computer Graphics & Virtual Reality*, pages 3–9, Las Vegas, Nevada, USA. CSREA Press.

Torpy, A. (2009). L3DT. Available from http://www.bundysoft.com/L3DT/.

Tutenel, T., Bidarra, R., Smelik, R., and de Kraker, K. J. (2008). The Role of Sematics in Games and Simulations. *ACM Computers in Entertainment*, 6:1–35.

Voss, R. F. (1985). *Fundamental Algorithms for Computer Graphics*, chapter Random Fractal Forgeries, pages 805–835. Springer-Verlag, Berlin.

Št'ava, O., Beneš, B., Brisbin, M., and Křivánek, J. (2008). Interactive terrain modeling using hydraulic erosion. In Gross, M. and James, D., editors, *Eurographics / SIGGRAPH Symposium on Computer Animation*, pages 201–210, Dublin, Ireland. Eurographics Association.

Watson, B., Müller, P., Veryovka, O., Fuller, A., Wonka, P., and Sexton, C. (2008). Procedural Urban Modeling in Practice. *IEEE Computer Graphics and Applications*, 28(3):18–26.

Weber, B., Müller, P., Wonka, P., and Gross, M. (2009). Interactive Geometric Simulation of 4D Cities. *Proceedings of Eurographics 2009*, 28:481–492.

Wonka, P., Wimmer, M., Sillion, F., and Ribarsky, W. (2003). Instant Architecture. In *SIGGRAPH '03: Proceedings of the* 30^{th} *Annual Conference on Computer Graphics and Interactive Techniques*, pages 669–677, New York, NY, USA. ACM.

Yong, L., Congfu, X., Zhigeng, P., and Yunhe, P. (2004). Semantic Modeling Project: Building Vernacular House of Southeast China. In *VRCAI '04: Proceedings of the 2004 ACM SIGGRAPH International Conference on Virtual Reality Continuum and its Applications in Industry*, pages 412–418, New York, NY, USA. ACM.

Zhou, H., Sun, J., Turk, G., and Rehg, J. (2007). Terrain Synthesis from Digital Elevation Models. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):834–848.

Massive Battle: Coordinated Movement of Autonomous Agents

Alberto Boccardo, Rosario De Chiara and Vittorio Scarano
ISISLab - Dipartimento di Informatica ed Applicazioni "R.M. Capocelli"
Università degli Studi di Salerno
{dechiara|vitsca}@dia.unisa.it

Abstract

The simulation of groups of characters moving in a virtual world is a topic that has been investigated since to 1980s. A widespread approach to this problem is the boid model. In the boid model agents, named boids, simulates the flight of a flock of birds. The keystone of the system is the concept of behavior: for each boid in the flock, a simple geometric calculation based on the positions of a limited number of boids, suggests an acceleration along a certain direction. Mixing more behaviors permits to create more complex behaviors like the boids following each others or avoiding obstacles. Starting from this, we present a system, Massive Battle, that, by extending basic behaviors, reproduces the complex movements of platoons of soldiers marching along a path and even simulates them engaging in a battle. The system is designed for online interactive simulations.

Keywords: Simulation, Autonomous Agents, Boid-model;

1 Introduction

The simulation of groups of characters moving in a virtual world is a topic that has been investigated since the 1980s. A widespread approach to this kind of simulations has been introduced in [4] and it takes inspiration from *particles system* [3]. In a particle system there is an emitter that generates a number of particles that move accordingly to a set of physics inspired parameters (e.g. initial velocity, gravity). The particle system approach is expanded with the purpose of simulating a group of more complex entities, dubbed *autonomous agents*, whose movements are related to social interactions among group members. A classical example of use of this approach is the simulation a flock of birds in the most natural possible way. Elements of this simulated flock are usually named *boids* (from *birdoid*) and got instilled a range of *behaviors* that induces some kind of *personality*. Actual implementations of such personalities can vary [14, 6], but the idea is the following: at every frame of the simulation, for every boid, for each behavior in the personality the system calculates a request to accelerate in a certain direction and the actual movement is the result of a weighted sum of all the contributions for a certain boid. The behaviors are, in the most of cases, simply geometric calculations that every boid makes, considering the nearest boids it is flying with: for example the behavior called *pursuit* just let the boid to pursuit a moving target (e.g. another boid). Boids react to their neighbors so they must be able to identify them by filtering nearby boids out of the whole population. The most simple way of do this filtering consists in a $O(n^2)$ proximity screening, and for this reason the efficiency of the implementation is yet to be considered an issue.

The number and the type of the behaviors allow to implement different personalities: a *leader* in the flock is a particular boid who has the knowledge of the path to follow. To differentiate a leader boid among the flock is pretty easy, while every boid in the flock has the behavior of following the leader, the leader has the behavior of following the path, ignoring other boids. In [1] is presented a study where such model is investigated and the simulated flock demonstrates the

capability of being able of following a path (e.g. toward food, along a migration route) even if the number of informed boids, that is leaders, is a very small proportion of the whole group.

In the following discussion we will refer to the *boid model* as the model presented in [4]. The boid model is designed for the aggregate motion of a simulated flock of boids as the result of the interactions of the relatively simple behaviors.

1.1 Previous works

In [8] is presented a technique to let a group of mechanical robots to navigate an environment with obstacles. The technique is based on potential functions: every point in the space has an assigned potential value that measures how much a robot is attracted by it. Robots will move around looking for the highest potential that usually is assigned to the goal to be reached. Obstacles will provide negative potential in order to repulse robots. The methodology is also extended by letting robots to coordinate in order to create a formation: every robots has a series of attachment sites that will attract other robots, in this way robots will just attach each others in various formation by defining suitable attachment sites.

The movement in an obstacles field is approached in [9], where it is presented a solution that plans the movement of a group of units without splitting it. The algorithm performs two phases: on the first phase it calculates the *backbone path* while in the second phase the actual group movement is performed. The backbone path is a function that describe a "corridor" amidst obstacles where the clearance at every point on the path is at least the radius of the enclosing circle of the units. The backbone path is the only area in the map where the units are allowed to walk during the second phase, where units actually walk along the path directed by a potential function. This solution clearly achieve the goal of letting the group to walk in an area avoiding splits.

A different solution to the problem of group splitting while traversing an obstacles filled area is presented in [10]. The paper offers two results: firstly an algorithm to control the steering behavior of groups based on a boundary value problem (BVP) path planner, then a strategy to handle the group formation-keeping problem effective to use any desirable formation shape. An interesting implementation detail of the techniques is that because the BVP path planner is based on the equation of Laplace, it is suitable to be efficiently implemented on GPUs and multi-core CPUs. The idea on which the path planning is based is a *group map* that surrounds the group during its movements on the map. The group map keeps track of the obstacles as repulsive areas within it, while the formation positions are attractive areas.

A commercial application that worths to be cited here is Massive [12], whose name is an acronym for Multiple Agent Simulation System in Virtual Environment. Massive allows to design, simulate and render complex scenes containing up-to millions of agents. The agents simulated personalities can other an high degree of realism. The system is not designed to render scenes in real-time.

This paper describes a system capable of animating autonomous agents with the purpose of reconstructing interactive scenes from a battlefield showing a number platoons fighting each others. Each platoon presents different soldier topology deploying different kinds of weapons. In this scenario we also liked to introduce a "courage" factor that let the user to tweak the ability of the soldier to stick to its purpose of killing the enemy. We mutated the idea of expanding previous models in order to reach an higher degree of complexity of the behaviors. We will use the boid model as the foundation on which to build more complex behaviors, the initial idea of simulating a flock of boids will be expanded to simulate a platoon of soldiers obeying to commands imparted by a leader. Soldiers not only will be able to march along a path but will also be capable of engaging a fight with enemy platoons.

The contribution of our paper are two successive steps of the process of expanding the boid model toward a more complex model. On the first step we show how by using basic behaviors we are able to convey to a group of agents the capability of marching like a platoon, executing directional commands. On the second step we extend the capabilities of executing orders to provide soldiers the capability of engaging with enemiy platoons, if any in sight. In Figure 1 is shown how

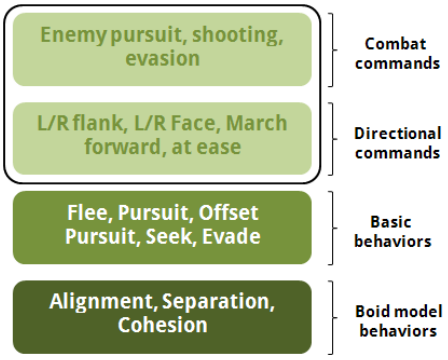


Figure 1: A vision of the presented technique.

the whole system is built: on the lower level there is the boid model, that offers basic behaviors to simulate a flock of boids; on the upper (second) level there are slightly more complex behaviors that are similar to boid model's behaviors, but offers different functionalities; both these level are implemented in [11] which is the library we used in our prototype. The upper levels in Figure 1, surrounded by a rectangle, represents the successive extension of functionalities offered by our system: the first one is the capabilities of marching as a platoon that is obtained by overlapping the effects of two of more basic behaviors; the other one is the combat capability, that is offered by slightly modifying the marching capability.

2 Coordinated movement

The target of animating units belonging to platoons in a realistic way has been inspired by real platoons parade in which soldier execute a certain amount of activities in a coordinated way. Soldiers usually are trained since first days of their carrier to march by coordinating just by looking at a small amount of their neighbors and, no matters of this limited coordination effort, they are able to follow orders fed them by the *commander*. A fundamental role is played by the *leader*, a special unit that in charge of to coordinate troops by the mean of specific directional commands.

In the following we will use the terms "soldier" and "unit" as synonyms. We will use the term "platoon" to mean an amount of soldiers within our system; this is an abuse and does not have any relation to real platoon, also considering the varying number of components in simulated platoons.

The Massive Battle system handles 4 different types of information that made up the scene that the user intends to simulate:

The map: the whole simulation will be carried out on a bidimensional map;

Obstacles: are static objects units have to outflank;

Leader: it is the commander of a platoon, it feeds the unit with directional commands;

Units: units made up a platoon and execute directional commands by just exploiting a local vision of the platoon.

All these different types of information are collected in a script that describes the map, the obstacles placed onto it, the number and the type of the platoon and for each platoon, it contains a description of the unit characteristics.

2.1 Basic behaviors

The whole system architecture is shown in Figure 2: on the upper level there is the script containing a path made by checkpoints, to reach each checkpoint, the leader will impart directional commands

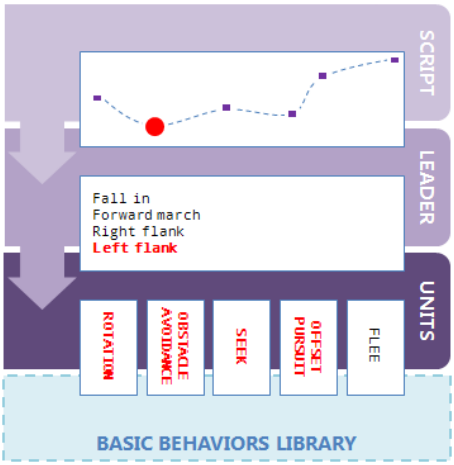


Figure 2: The information exchange between the three level of command.

to the troops, every directional command will be translated in a set of basic behaviors implemented by a library of basic behaviors. The example in Figure 2 shows what happens on the checkpoint shown in boldface, the directional command that will be imparted by the leader and the basic behaviors that will be taken into account by units.

We will shortly summarize a set of basic behaviors the system uses to translate high level directional commands received by the leader.

Seek this behavior lets the unit (agent) to move toward a point; The Flee behavior is the opposite of the Seek behavior: the unit runs away from a certain point. In Figure 3.(a) the effects of the Seek and the Flee behaviors are shown.

Pursuit it is similar to the Seek behavior but the fact that it applies to a moving point (e.g. another agent). This behavior takes into account a prediction of the movement of the point to follow. The Evade lets an agent to run away from a moving point. In Figure 3.(b) the Evasion and Pursuit behaviors are compared. A slightly different behavior is the Offset pursuit behavior that lets a unit to pursuit another unit keeping a certain distance (offset) between them.

Obstacle Avoidance this is a fundamental behavior in which the obstacles present in the scene are took into account. In Figure 3.(c) is shown how among 3 obstacles (A, B, C) just the nearest one (B) is taken into account (red arrow).

Flocking this behavior is the one directly inspired by looking at real bird flying in formation. Even if it can be considered as a basic behavior, in the boids-model it is the result of three behaviors that are applied simultaneously Separation, Cohesion and Alignment. Separation lets a boid to stay away from its neighborhood and Cohesion is the opposite of this behavior. Alignment lets the boid to align its fly direction to the average of neighborhood boids directions.

2.2 The leader

The leader mimics the role of the commander of a real platoon of soldiers. The commander usually marches besides the platoon and feeds the platoon with commands. Following a list of the implemented commands [2]:

Fall in: soldiers get together and form a platoon;

Forward march: soldiers start marching in the direction they are facing;

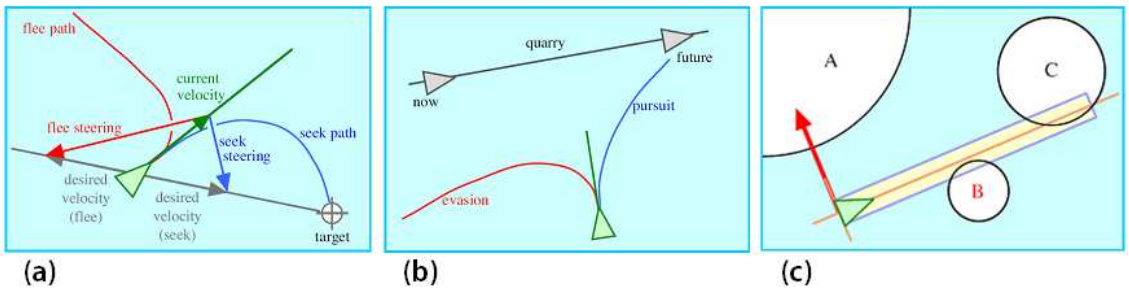


Figure 3: Different basic behaviors: (a) Flee and Seek; (B) Evasion and Pursuit; (C) Obstascle avoidance (Figure from [5]).

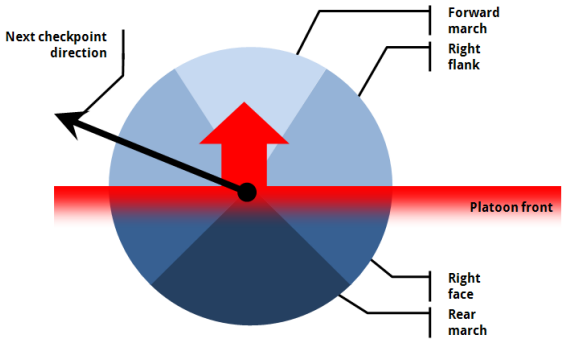


Figure 4: How the leader chooses the correct command to conduct the platoon to the next checkpoint (rightward commands shown). In this example the leader will choose to impart a Left Flank command to the platoon.

Rear march: the platoon performs a 180 degrees turn, while marching;

Right (Left) flank: the platoon performs a 90 degree pivot to the right (left), while marching.

Halt: soldiers stop the marching immediately;

Right (Left) face: a command given from a halt to turn 90 degrees to the right (left);

To march at ease: soldiers can march avoiding keeping in step because of the rough terrain.

The simulation consists of a certain number of platoons together with their paths to be followed. Paths, made by checkpoints, are followed automatically by every platoon. The platoon commander follows every checkpoint it finds along the path, one by one, feeding suitable commands to the troops. At every checkpoint the suitable command is simply chosen among the directional commands in the list above, by using the angle between the current direction and the direction of the next checkpoint. In Figure 4 is shown how this selection is implemented.

2.3 Assembling behaviors

To perform the directional commands described in previous section different units in the platoon must play different roles: this is exactly what happens in reality.

Fall in command. The fall in command is sent to every unit in the platoon. Every unit has an associated position within the platoon and reach this position by assuming an offset pursuit behavior. Once the unit reached its final position it notifies the leader, and whenever the fall in is completed, the commander will issue the next command.

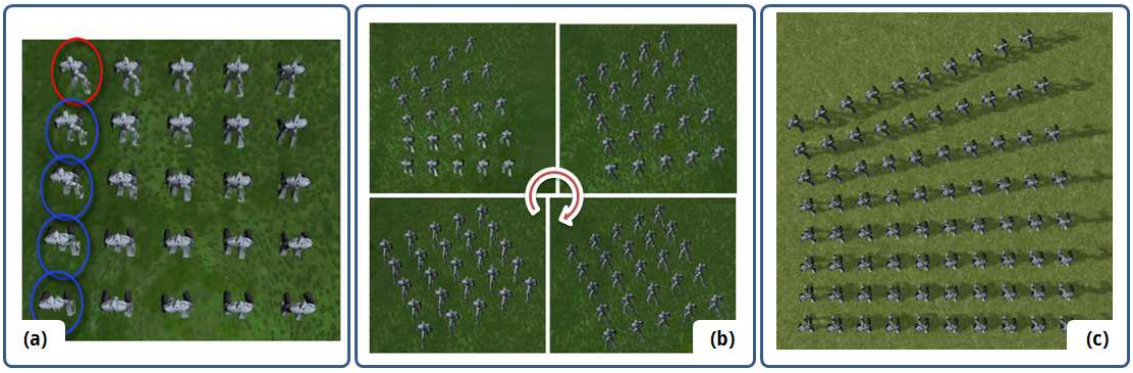


Figure 5: The left flanking (a) different roles played by soldiers: red circle the platoon pivot, blue circles the line pivot; (b) the result of a left flanking (from upper left, clockwise); (c) the same command execute by a larger platoon.

Left flank command. In Figure 5.(a) the three different roles played by soldiers to perform a left flank: the leftmost soldier on the front line is the *platoon pivot*, the leftmost soldiers on each line are the *line pivots* and the rest of non-pivot soldiers keep on marching by coordinating each others during the flanking. The differences between the roles are implemented by using a number of basic behaviors. The formation pivot performs a simple rotation by marching in place; line pivots will keep the rotation together with a seek behavior on the formation pivot; the other soldiers will perform a simple offset pursuit on the line pivot of the the line they belongs to. In Figure 5.(b) is shown, starting from the upper left figure and going clockwise, the effect of the left flanking of a 25 units platoon.

Marching at ease. When an obstacle gets in sight of a unit it communicates this information to the leader that will impart the command of Marching at ease, this will let soldiers to keep moving toward the checkpoint without marching in formation. This high level behavior is obtained by assembling the effects of three basic behaviors: obstacle avoidance to circumvent obstacles, seek toward the checkpoint and flocking to not to get too far from others.

2.4 Combat

Once the platoon is able to walk along a path executing the directional commands imparted by the leader, is reasonable to model the situation in which two or more adversary platoons come insight. We have took into account this situation by firstly offer three different models of weapons: melee weapon model, mortar weapon model and rifle weapon model. Each platoon within the simulation has an identification that is used to discriminate among friend and foe platoons. While the leader is conducting the platoon along the checkpoints it always queries the system in order to verify if an enemy is in sight. Once an enemy is visible by the leader, the leader switches to *combat mode* and impart the same command to the soldiers.

Before describing the details of the combat mode we shortly summarize the differences running between the three weapons model available in the system. A *melee weapon* is a model that is used just in the clinch fighting, so to be effectively used, soldiers have to reach the enemy at a very short range. A *mortar weapon* simulates the behavior of a weapon shooting bullets with a high-arcing ballistic trajectories. The *rifle weapon model* produces bullets that runs horizontally to ground, 6.(a).

For each weapon model the system offers a wide range of parameters describing different aspects of how the weapon is actually simulated. The system not only simulates the trajectory of mortar-like shooting, but also allows the user to define the typology of bullets shot, by choosing the radius of the explosion induced by the impact of the bullet with the ground (see Figure6.(b)). The configuration of the weapons is pretty flexible, for example, by reducing the impact radius

of the mortar bullet is possible to simulate an arrow shot by a bow. For the rifle weapon model is possible to define the perforation capability, that is, the capability of trespassing through a number of bodies. In Figure 6.(c) is shown the effect a perforation value set to 2. A common trait of the three different models is the range of action the enemy must be within, in order to let the soldier to effectively use it.

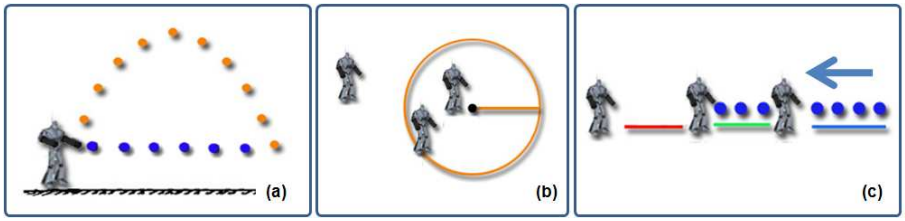


Figure 6: Simulated shooting characteristics (a) comparison between the mortar shooting model and rifle shooting model; (b) the explosion radius of the mortar bullet; (c) the perforation capability of rifle bullets.

When the leader is in combat mode it will keep track of the enemy platoon position and will feed it to the soldiers. Once the soldiers receive the command of switching to combat mode they will stop marching in formation and start marching at ease toward the checkpoint assigned by the leader, that is the position of the enemy. Each soldier will choose an enemy soldier it will keep walking toward until the enemy will gets in the weapon range. Once the weapon can be used to effectively hit the enemy, soldier will try shooting, and will keep doing so until the enemy dies, it will run away from the shooting range or will run out of bullets.

The described combat mode is implemented by using three additional behaviors depending on a factor called *courage*: the courage is used to decide to whether attack a soldier or not. Let consider 2 soldiers involved in a potential fight and their the courage factor, three cases can occur: if the courage of the attacking soldier is low it runs away from the enemy by assume an Evade behavior. If the enemy choose to evade, the soldier will pursuit it, by assuming a Pursuit behavior and shooting whenever the enemy is in range. If none of the soldiers evade, they will engage fighting, by assuming the Seek behavior.

The combat fighting is just a proof-of-concept of the idea of extending the marching behaviors by offering slightly different functionalities.

3 Conclusion

From a technological point of view we developed Massive Battle as a framework, written in C++, that is capable of interpreting a script file. The script file contains a full description of the initial setting of the parameters for each platoon. Once a file is loaded the simulation starts and the user can just sit and enjoy what happens on the screen. The system is implemented in a library that does not depend on the graphics engine used to represent the units. This widen up the scenarios of use: for instance, even if the library is designed for real time operations, it could be used to generate realistic movements of troops for high quality off-line rendering program.

Massive Battle is tested within an environment that simulates a battle among a number of soldiers grouped in platoons in a 3D scenario. Our system has been designed as a rapid prototype system to reconstruct famous historical battles. The basic simulation functionalities can be used to investigate different options, for example once the simulation is reproducing a real battle (e.g. the La Haye Sainte battle), the system allows the user to verify how the history could be different by just varying some of the settings (e.g. the heavy cavalry brigades attack earlier). We have also implemented a *Battle Editor* that provides all the necessary editing tools to modify the the morphology of the terrain and obstacles (e.g. rivers, mountains, buildings etc. . .). Together with the editing tools, the system also offers a set of tools to enhance the fruition of the simulation: it is possible to choose the speed of the simulation, to change the camera position and to stick the camera on a certain soldier.

Figure 7: Two screenshots from the application: (left) 6 platoons confrontating; (right) follow-up camera.

together with some validation performed by historical researchers.

The graphical engine we used to implement the visual part of the system is Ogre3D [13]. The basic behaviors library we used is presented in [11], but similar behaviors can be found also in the OpenSteer library [7]. It is worth mentioning some details about the performances of the system on a off-the-shelf PC: AMD Athlon 64 x2 4200+, 2GB of RAM, ATI X1900 with 512MB. The system was able to animate 3000 units on an interactive framerate of 25 fps (frames-per-second). This performances suggest an avenue for future researches, that is, implementing the behaviors on parallel architecture like GPUs or Multicore CPUs, in particular the spatial database used to query about positions of the agents.

References

[1] Couzin ID, Krause J, Franks NR, Levin SA. Nature. 2005 Feb 3;433(7025):513-6. Effective leadership and decision-making in animal groups on the move.

[2] USA Marine Corps Drill and Ceremonies Manual MCO P5060.20

[3] Reeves, W., T., "Particle Systems-A Technique for Modeling a Class of Fuzzy Objects", ACM Transactions on Graphics, V2–2, April 1983. and reprinted in Computer Graphics. V17–3, July 1983, (acm SIGGRAPH '83 Proceedings), pp. 359-376.

[4] Reynolds C. Flocks, herds and schools: a distributed behavioral model. In SIGGRAPH'87: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, ACM, New York, NY, USA, 1987.

[5] Reynolds C. Steering behaviors for autonomous characters. In Game Developers Conference, Miller Freeman Game Group, San Francisco, CA, USA, 1999.

[6] Reynolds C. Big fast crowds on PS3. In Sandbox'06: Proceedings of the 2006 ACM SIGGRAPH Symposium on Videogames, ACM, New York, NY, USA, 2006.

[7] http://opensteer.sourceforge.net/

[8] Balch T, Hybinette M. Social potentials for scalable multirobot formations. In IEEE International Conference on Robotics and Automation (ICRA 2000), San Francisco, 2000.

[9] Kamphuis A., Overmars M. H. Motion planning for coherent groups of entities. In IEEE Int. Conf. on Robotics and Automation. IEEE Press, San Diego, CA, 2004.

[10] Silveira, R., Prestes, E., and Nedel, L. P. 2008. Managing coherent groups. Comput. Animat. Virtual Worlds 19, 3-4 (Sep. 2008), 295-305.

[11] Buckland M. Programming Game AI by Example. Wordware Publishing, 2005.

[12] Massive software `http://www.massivesoftware.com`. Accessed on May 2009.

[13] Pro OGRE 3D Programming, (Gregory Junker).

[14] R. De Chiara, U. Erra, M. Tatafiore and V. Scarano. Massive simulation using GPU of a distributed behavioral model of a flock with obstacle avoidance. Proceedings of Vision, Modeling, and Visualization 2004 (VMV 2004) (Stanford - California, USA, Nov 16 - 18, 2004). pp. 233-240.

Detection Tracking and Recognition of Human Poses for a Real Time Spatial Game

Feifei Huo, Emile A. Hendriks, A.H.J. Oomes
Delft University of Technology
The Netherlands
f.huo@tudelft.nl

Pascal van Beek, Remco Veltkamp
Utrecht University
The Netherlands
phbeek@students.cs.uu.nl

Abstract

In this paper, we present an approach to detect, track people, and recognize poses. The detected poses are used for controlling a real time spatial game. In the people detection, tracking and pose recognition system, body parts such as the torso and the hands are segmented from the whole body and tracked over time. The 2D coordinates of these body parts are used as the input of a pose recognition system. By transferring distance and angles between the torso center and the hands into classifier feature space, simple classifiers, such as the nearest mean classifier, are sufficient for recognizing predefined key poses. The output of the classifier, that is the identification of the pose, is used to control color and actions of the virtual actor. The position of the virtual actor is steered by the detected position of the user in the image.

Keywords: Pose Recognition, Spatial Game, Real Time.

1 INTRODUCTION

Nowadays video-based applications have become more and more widespread [1]. A well-known video-based application is man-machine interaction, in which people can use their facial expressions, gestures and poses to control e.g. virtual actors or (serious) games. The essential ingredient for an effective man-machine interaction experience is that the system indicates its level of understanding of the user's movement. Therefore, human motion analysis plays an important role in man-machine interaction. Generally, there are two approaches to obtain the movement of human body. One approach is marker-based, in which users need to wear specific suit with sensors on it. These sensors are used to capture the motion of different body parts. The other approach is vision-based, in which users are totally free of any obtrusive sensors. The movement of users is analyzed from the recorded video data. Compared with the first approach, the second one may have less accuracy of reading motion information. However, it is more convenient and friendly to users, especially for gaming applications. Therefore in this paper, we propose a vision-based people detection, tracking, and pose recognition system. It directly uses the captured video frame as input, then gives the 2D position and pose of the people if there are people appearing in the scene. The position and pose information is connected to a spatial game system, and used as the control command of the spatial game. The remainder of the paper is organized as follows. In Section 2, we give a brief introduction of previous researches. The methodology of the proposed approach is described in Section 3. In Section 4, we show the spatial game application. At the end, the conclusion is drawn in Section 5.

2 PREVIOUS RESEARCHES

Although there has been published a significant number of research papers on human body tracking and pose recognition, many research issues still remain to be solved. In [2] the body parts are reliably labeled and located by 2D contour shape analysis. Tracking performance is significantly increased by taking color into account. The limitation of this method is that all the body parts need to be segmented although they may not be needed for certain applications. In [3], an approach to estimate 3D human pose with multiple cameras is proposed. It can deal with cluttered scenes and self-occlusion under some constrains. But the processing time for pose estimation is about 45

seconds per frame, which is not suitable for real-time applications. In [4] an exemplar based approach is used to localize and recognize human poses. However, the pose detector is only learnt from walking poses, so they can not detect people with other poses than walking. In this paper, we present an approach for real-time people detection, tracking, and pose recognition, which can handle a variety of poses and is fast enough to steer a real time game. The output of the pose classifier is used for controlling appearance and actions within this spatial game. Fig.1 gives the flowchart of the proposed system.

Fig.1. The flowchart of the proposed system.

3 METHODOLOGY

3.1. MOTION EXTRACTION

In an indoor scenario, cameras are usually at fixed locations, so we do not need to consider motion of the scene for foreground object extraction. Therefore, background subtraction is a quite suitable method in this case. Compared to temporal differencing, the background subtraction will not only give the edges but the whole silhouette of the moving objects. The first step of the background subtraction is to build up a background image, which should not include any foreground objects. In our implementation, the background image is built by using a mixture of Gaussian model in [5]. This method is also robust to cluttered scenes. The background image is updated by using current frames, in order to deal with the change of lighting conditions. After the background image is obtained, the pixel-wise difference between the current frame and the background model is used to classify each pixel as either foreground or not. A difference image $D_{i,j}^t$ between an input image $F_{i,j}^t$ and the background image $B_{i,j}^t$ shows only the moving object regions, while the stationary background is suppressed (see eq.1).

$$D_{i,j}^t = F_{i,j}^t - B_{i,j}^t \tag{1}$$

A foreground binary image is obtained by using the difference image $D_{i,j}^t$ and a threshold T_d (see eq.2). T_d can be obtained experimentally and is found out to be not very sensitive for different indoor scenes. Fig.2 shows the input image $F_{i,j}^t$ and its foreground binary image $R_{i,j}^t$.

$$R_{i,j}^t = \begin{cases} 1 & if\ D_{i,j}^t \geq T_d \\ 0 & else \end{cases} \tag{2}$$

(a) (b)

Fig.2. Extraction of foreground binary image: (a) input image $F_{i,j}^t$, (b) foreground binary image $R_{i,j}^t$.

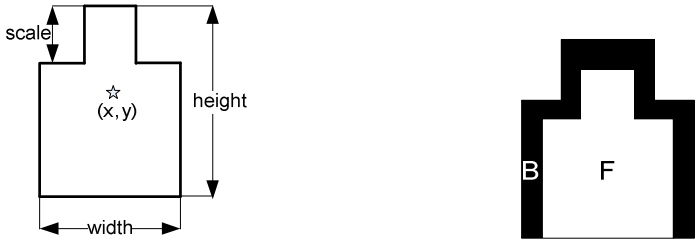
3.2. TORSO AND HAND SEGMENTATION

After we obtain the motion-based human body blob, next step is to segment different body parts. This can be based on the selection of various features, such as shape, edge, silhouette, contour and color of human's body. We use a 2D silhouette model to detect the torso and use skin color for the detection of hands.

3.2.1 Torso segmentation

After the foreground binary image is built, we use a geometrical characteristic of persons to detect them. The prior knowledge of human geometrical structure that we use is a head-shoulder-upperbody model, shown in Fig.3. The parameters in this 2D human shape model are position and scale, so this model is described as $P = (x, y, scale)$. If the scale parameter in the 2D model is determined, the width and length of human upper body can also be derived according to the shape characteristic of persons. Since the recorded video data is from one single camera, the position of the human is given by 2D coordinates, that is x and y coordinate. In the real 3D world, this method can be easily extended to multiple view approaches by fusing the 2D data derived from synchronized cameras into 3D real coordinates.

After the definition of the human model, the problem comes up with how to use this 2D model to determine if there is a person in the image or not. Generally the intuitive solution is exhaustively searching the image for the defined model. Although there are only three parameters in this model, x y coordinate and scale, it still needs a large amount of calculations to find the global optimal solution. In the situation of more than one person appearing in the image simultaneously, this exhaustive search is too far away from real time applications. So it is impractical to use this model directly for template matching. However, this detection and tracking problem can be seen as a state estimation problem. To solve it, statistical methods, such as particle filters, can be used, which is especially suitable for non-Gaussian and multi-model situations. In a particle filter, the probability of detection of a person in the image is represented as the fitness coefficient of the defined 2D shape model. The definition of this fitness coefficient is the same as [6]. The template used to calculate the fitness coefficient is shown in Fig.3 (b). It is composed of two regions: foreground region F and background region B, which is surrounding region F.



(a) (b)

Fig.3. (a) One sample in a particle filter, (b) two dimensional human model to calculate fitness coefficient [6].

In order to construct the probability, the fitness coefficient should be a value between 1 and 0. This requirement is satisfied by using the following definition (see eq.3).

$$\omega = \frac{1}{Area(F)} \times \left\{ \begin{array}{ll} \sum F - \sum B, & if \quad \sum F > \sum B \\ 0, & \text{otherwise} \end{array} \right\} \tag{3}$$

Where $Area(F) = Area(B)$, $\sum F$ is the summation of the pixel values in region F, $\sum B$ is a summation of the pixel values in region B. Hopefully foreground pixels with a value 1 will fall into region F as much as possible and background pixels with a value 0 are only included in region B. Under this optimal situation the fitness coefficient will be 1, which means the foreground binary image satisfy the 2D model perfectly. We use this fitness coefficient ω as the probability of a person present in the image. The larger the value of the fitness coefficient is, the higher the probability of a person existing in the image. By fitting this 2D model on the foreground binary image, people's head and torso can be segmented from other parts.

3.2.2 Hand segmentation

In addition to the 2D model mentioned in 3.2.1 for human's torso detection and tracking, foreground pixels are further segmented into skin-color and non-skin-color regions. A skin color model in the RGB color-space is used to select skin color pixels on the foreground image. This human skin color model is similar to the model in [7]. If foreground pixels mapping into the RGB color-space satisfy the following conditions in eq.4 [7], they will be considered as skin-color pixels. People's face and torso region are excluded from skin color detection by masking the head and torso region estimated from 3.2.1.

$$\left|\arctan(\frac{B}{R})-\frac{\pi}{4}\right|<\frac{\pi}{8}, \qquad \left|\arctan(\frac{G}{R})-\frac{\pi}{6}\right|<\frac{\pi}{18}, \qquad \left|\arctan(\frac{B}{G})-\frac{\pi}{5}\right|<\frac{\pi}{15} \tag{4}$$

After the skin color pixels are selected, two post-processing steps are used to get rid of false positive detections. The first step is to delete regions with a very small size, which are impossible to be hand regions. In the second step, a motion mask is introduced to exclude regions which are far away from previous hand locations. It limits the movement of hands within a certain bounding box. From the remaining two largest blobs, we calculate the centers of gravity and use them to represent the position of the hands.

3.3. FEATURE SPACE CONSTRUCTION

The input of the proposed pose recognition system are 2D positions of the torso center and the hands. However, we transfer them into normalized feature space and train the classifier in this new feature space. The reason is that the pose recognition system should be scene invariant. That is, no matter where the person is in the scene, or how far the person is from the cameras, the predefined key poses should be recognized. Therefore the feature space is built by using angles and relative positions between hands and torso center. We construct the following 6 feature components, denoted as $F_{set}=\{c_1,c_2,c_3,\ldots,c_6\}$:

$$c_1=\frac{(x_2^l-x_2^t)}{s},\ c_2=\frac{(y_2^l-y_2^t)}{s},\ c_3=\frac{(x_2^r-x_2^t)}{s},\ c_4=\frac{(y_2^r-y_2^t)}{s},\ c_5=\arctan\frac{y_2^l-y_2^t}{x_2^l-x_2^t},\ c_6=\arctan\frac{y_2^r-y_2^t}{x_2^r-x_2^t} \tag{5}$$

Here (x_2^t,y_2^t), (x_2^l,y_2^l) and (x_2^r,y_2^r) are the 2D positions of the torso center, left hand and right hand. s is the scale parameter in 2D model. The classifier will be trained and tested on this 6D feature space F_{set}.

3.4. POSE CLASSIFICATION

The key poses are designed for gaming control, so they should be easy for users to remember and perform. We also choose the number of the poses not too high to make it easier for users. In our system, we defined nine poses in total, as shown in Fig.4. From top row to bottom row and left to right, these nine poses are labeled as pose1 to pose9.

In order to build a classifier, we manually labeled the frames containing the nine poses into nine classes. For each pose, the samples are selected from different persons. Our experimental data set contains 1515 samples of 9 pose types (classes) and 6 features. On average, each pose class is represented by 170 samples. The performances of several statistical classifiers with different complexities are compared. Specifically, we evaluated the nearest mean classifier (NMC), the linear classifier (LDC) and the quadratic classifier (QDC) assuming normal densities and the non-parametric Parzen classifier [8]. We observe that the simplest method (NMC) provides comparable performance to more complex classifiers which need an extra dimensionality reduction step to avoid the curse of dimensionality. We conclude that the extracted features are informative and do not require use of more complex classifiers. Therefore, we chose a simple classifier, 10-nearest neighbourhood classifier (NNC), for the pose classification part of our system. It is easy to implement and also benefits from the computation point of view.

Fig.4. Predefined key poses. From top row to bottom row and left to right, these nine poses are labeled as pose1 to pose9.

3.5. RESULTS AND DISCUSSION

The proposed system is implemented in C++ with OpenCV libraries [9]. The processing time for each frame is 0.047 seconds, including background subtraction, body parts segmentation and pose recognition. We tested the system with more than 20 users. We also chose the different indoor environment with various settings. Some of the experimental results are shown in Fig.5. An online video demo is also available [10]. People's head and torso are indicated with yellow and red rectangles. Within this 2D model, the location of head top, head center, torso center, torso bottom and both shoulder can be estimated, which are presented by yellow and red cross. People's left and right hand are marked with blue and green cross separately. The output of the pose recognition system is an integer, the number shown in Fig.5. It gives the indication which pose user is performing. This integer and the 1D (horizontal) position of the user will be used as the control command of a spatial game. We evaluated pose classifiers using cross-validation approaches. The average error for all the poses is 6% by using NMC. The result shows that there is a clear separation between pre-defined poses. We also calculate the confusion matrices of the 9-class pose classifier (NMC). The results are promising. Most of the poses can be recognized very well. More details about the pose classification can be found in [11].

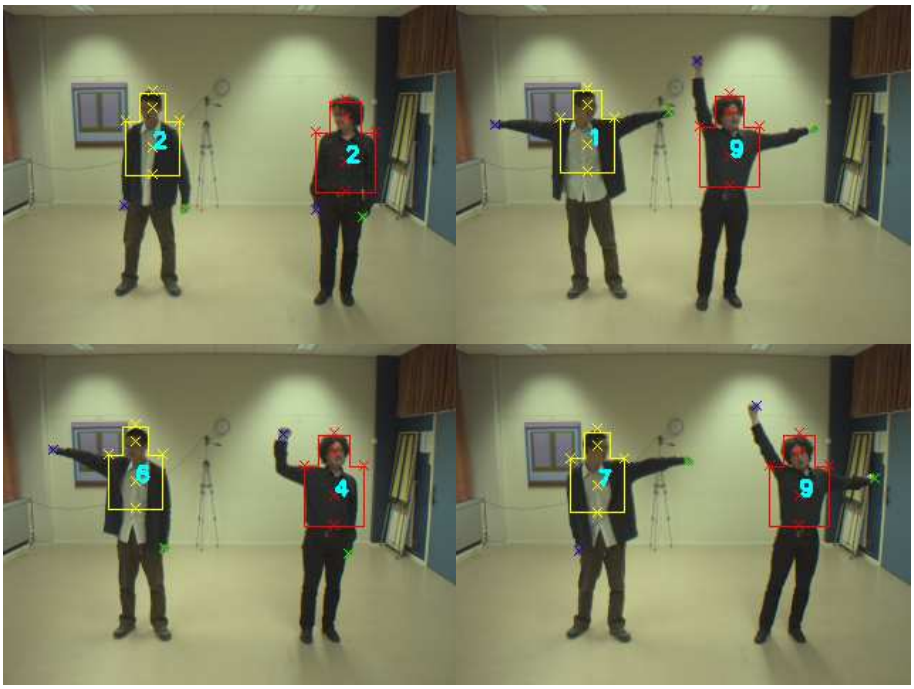
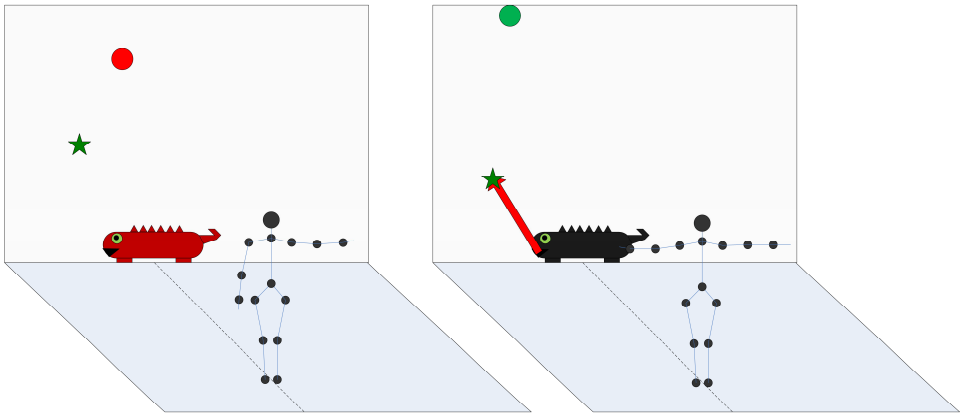


Fig.5. Results from people segmentation and pose recognition.

4. SPATIAL GAME APPLICATION

4.1. IMPLEMENTATION

As an application for the pose recognition system we implemented a spatial game, based on the proposal of Phong in [12]. This is a variation of the game Pong [13] in which the player controls a bat to bounce off balls. In Phong the player controls a chameleon which has to bounce off photons, see in Fig.6. The position of the chameleon is determined by the player's position in front of the camera. The photons can have 6 different colors: red, blue, green, yellow, cyan and magenta. The chameleon can change into each of these colors when the player adopts the appropriate pose.



(a) (b)

Fig.6. (a) Color and position of the chameleon are controlled by pose and position of the player, (b) the tongue of the chameleon is also controlled by a pose to catch flies [12].

When the photon hits the ceiling, it changes color. When the photon is bounced off while the chameleon has the wrong color, the controls flip. Left becomes right and vice versa. When the chameleon has the right color while bouncing the photon off, the score and speed of the photon is increased. When the chameleon misses the photon the ground is heated up. After 4 misses the ground is too hot and the game is over. At random moments a bug flies into the scene which can be eaten by the chameleon when the player adapts to the eating pose. This will increase the score and the ground will cool down.

The game is implemented using the graphics engine Ogre [14]. The input for the game is given by the pose recognition system. The pose recognition system and the spatial game are two separate applications which communicate via sockets [15]. Therefore, it is possible for the two applications to run on different computers and communicate through a network. The pose recognition system sends two types of data to the spatial game in every time step. It sends an integer that represents a pose (1-9) and an integer representing the 1D-location of the player. Whenever the spatial game receives this data, it updates the position of the chameleon according to the position integer and it carries out the action belonging to the pose index that was send. These actions consist of 6 poses for changing the chameleon into the 6 different colors, 1 pose is for eating the bug, 1 pose is for starting the game and 1 pose is to pause the game. Fig.7 gives a screen shot of user playing the game. On the left side is the interface of the game, which shows the level, bounces, heat and score of the player. The three windows on the right side are the results from vision-based analysis. From top to bottom, they are original image, results from body parts segmentation and pose recognition, and foreground binary image.

4.2. RESULTS AND DISCUSSION

In our first test runs it became clear that the sensitivity of the pose recognition to detect the change of poses gave a problem for the game player. Whenever the player needs to change from one pose to another there could be a different pose adopted that is “in between” these two poses. When this happens the color of the chameleon in the game is shortly changed into an unwanted color. This problem has been overcome by using a counter whenever a new pose is adopted. The new pose has to be adopted for 4 consecutive time steps until its corresponding action is carried out. This adjustment improved the playability of the game as the user feels having a better control of the chameleon. We did encounter a short delay in handling the players input. The delay is caused by the image processing time. This is mostly noticed with updating the chameleon’s position by the player’s actual location, but the delay is too small to actually cause gameplay problems.

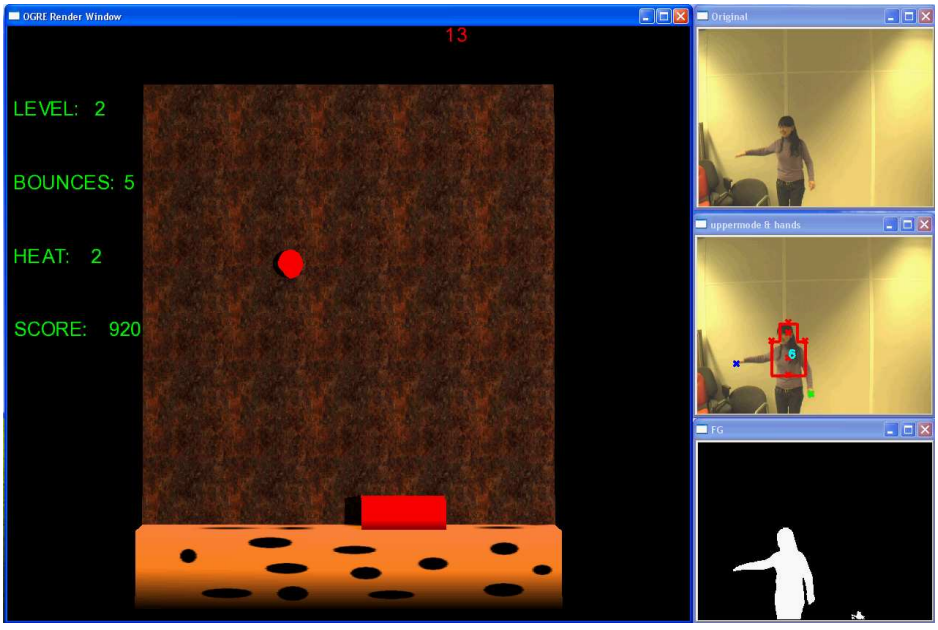


Fig .7. Spatial game interface. On the left side is the interface of the game, which shows the level, bounces, heat and score of the player. The three windows on the right side are the results from vision-based analysis. From top to bottom, they are original image, results from body parts segmentation and pose recognition, and foreground binary image.

The implementation of the Phong game showed that the gameplay of the spatial game is interesting. As a next step it is good to reduce the delay to a minimum. After this improvement it is interesting to create a more complex game with an elaborate user interface.

5 CONCLUSION

In this paper, we described a real-time computer vision based application. The proposed system is composed of two parts. The first part is video-based people detection, tracking and pose recognition system. It directly uses the captured video frame as input, then gives the 2D position and pose of the people if there are people appearing in the scene. The position and pose information is connected to the second part, a spatial game system, and used as the control command of the game. This pose-driven spatial game is a real time man-machine interaction without obtrusive sensors. It shows the possibility of a new way of interactions in novel computer games and entertainment. The combination of computer vision research and a practical application is quite useful. It allows us to directly test if the proposed algorithm satisfies certain requirements, in a specific application environment. Future work will include improving the robustness of the system (e.g. better skin color detection, more robust feature detection) and developing multiple-user applications. One of the challenges will be to solve the occlusion problem if users are allowed to move freely.

ACKNOWLEDGMENTS

This research has been supported by the GATE (Game Research for Training and Entertainment) project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie). The spatial game is based on the proposal of Berend Berendsen.

REFERENCES

[1] Moeslund, T. B. Hilton, A. and Kruger, V. (2006). A survey of advances in vision-based human motion capture and analysis. In *Computer Vision and Image Understanding*, vol. 104, pages 90-126, 2006.

[2] Wren, C. Azarbayejani, A. Darrell, T. and Pentland, P. (1997). Pfinder: real-time tracking of the human body. In *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol.19, no.7, pages 780-785, 1997.

[3] Gupta, A. Mittal, A. Davis, L. S. (2008). Constraint integration for efficient multiview pose estimation with self-occlusion. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 493-506, 2008.

[4] Rogez, G. Rihan, J. Ramalingam S. and etc, (2008). Randomized trees for human pose detection. In *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, June 2008.

[5] Stauffer, C. and Grimson, W. (1999). Adaptive background mixture models for real-time tracking. In *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. II, pages 246-252, 1999.

[6] Micilotta, A. (2005). Detection and tracking of humans for visual interaction. In *PhD. Dissertation, School of Electronics and Physical Sciences, University of Surrey*, 2005.

[7] Porikli, F.M. and Tuzel, O. (2003). Human body tracking by adaptive background models and mean-shift analysis. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2003.

[8] PRTools toolbox http://prtools.org and PRSD Studio http://prsdstudio.com software packages.

[9] http://opencv.willowgarage.com/wiki/

[10] http://prsysdesign.net/index.php/html/blog_comments/embedding_classifi

[11] Huo, F. Hendriks, E.A. Paclik, P. and Oomes, A.H.J. (2009). Markerless human motion capture and pose recognition. In *International Workshop on Image Analysis for Multimedia Interactive Services* (WIAMIS), 2009.

[12] Berendsen, B. (2008). Tracking and 3D body model fitting using multiple cameras. Master's thesis, UU, Department of Computer Science, INF/SCR-2007-066, 2008.

[13] http://www.pong-story.com/rhbaer.htm

[14] http://www.ogre3d.org/

[15] http://en.wikipedia.org/wiki/Internet_socket

The Future of 3D and Knowledge Technologies in Gaming and Simulation (FOCUS K3D position statement)

Wolfgang Hürst (Utrecht University, NL), Michela Mortara (IMATI, Italy), Chiara Eva Catalano (IMATI, Italy), Patrick Salamin (EPFL, Switzerland), Ewald Quak (SINTEF, Norway), Marco Attene (IMATI, Italy)

Synopsis. The following documents summarize some of the work that has been done in the Application Working Group (AWG) "Gaming & Simulation" of the FOCUS K3D project. FOCUS K3D is a Coordination Action in the European Union's 7th Framework Programme aiming the promotion of best-practices for the use of semantics in 3D content modelling and processing. We start with a short introduction describing some of the aims of FOCUS K3D and the AWG Gaming and Simulation. Then, we present a summary of a State-of-the-art-report on 3D content in Gaming and Simulation that was produced as part of this project. We present a description of the AIM@SHAPE Digital Shape Workbench – a powerful infrastructure for integrating, combining, adapting, and enhancing existing and new software tools and shape databases, and a concrete example demonstrating how current, state-of-the-art semantic processing tools can be used in a beneficial way in game development. The closing section presents position statements that evolved from the work done so far in FOCUS K3D and which we like to put up for discussion with the audience at the 3AMIGAS workshop.

Content

1. FOCUS K3D and the Application Working Group “Gaming & Simulation”

Wolfgang Hürst (Utrecht University, NL)

FOCUS K3D (ICT-2007-214993) is a Coordination Action (CA) of the European Union's 7th Framework Programme. It aims at promoting the adoption of best practices for the use of semantics in 3D content modelling and processing. For this, FOCUS K3D has identified four application areas that are both consolidated in the massive use of 3D digital resources (*Medicine & Bioinformatics* and *CAD/CAE & Virtual Product Modelling*) and emerging (*Gaming & Simulation* and *Archaeology & Cultural Heritage*). One of the major efforts in the CA’s promotion activities was the production of a state-of-the-art report (STAR) for each of the four application areas. These four STARs are meant to identify the general trends in the respective area, for which 3D-related semantic research needs to be pursued. In the remainder of the project – through thematic workshops, personal communications and the compilation of a research roadmap – the FOCUS K3D team will try to identify concrete steps to be taken in the near future that are heading in the right directions as suggested by the STARs.

The separation into different application areas was motivated by two reasons. First, each of these domains has different requirements and characteristics, and thus uses specific tools and methodologies. Identifying them, and highlighting the differences and similarities among the different areas is one of the major goals of FOCUS K3D. Second, there are also significant differences in the involved communities, especially considering the industrial partners. These are not only based on the actual applications but also, for examples, on the roots of a field and how it developed historically. The four STARs are therefore different in nature insofar as they reflect a snapshot of the current situation in four subfields that are bounded by the massive use of 3D data but distinct in their characteristics, requirements, status, and condition.

The *Gaming and Simulation* domain as well as the involved industries and community at large differ significantly from the other three areas that are addressed within FOCUS K3D. First of all, Gaming and Simulation is a rather new research field. This is not only true for the topic of “3D and Knowledge Technologies” tackled by FOCUS K3D but gaming-related research in general. Whereas video and computer games have become popular since the 80s, it was not until a few years ago that game development started to be considered an academic activity. Thus, unlike for example CAD and Medicine / Bioinformatics, there is no long tradition of research in the gaming industries. Even today, research activities of related companies are very limited. In addition, gaming is a very fragmented branch of industry. Running a successful company with less than ten employees is not uncommon. Similar to other application areas, there are high requirements for realistic 3D modelling because consumers of computer and video games are normally not willing to accept unrealistic, imperfect animations. This need for perfection is contradicted however by the high pressure existing in this market and tight production timelines that result from the constant desire of consumers to obtain the latest, state-of-the-art technology as well as fast changing trends in game play and covered topics.

These characteristics create a very unique situation. Unlike other application areas, such as CAD and Medicine where research and commercial applications are tightly bounded and developed hand in hand, scepticism against gaming research is not uncommon – even in the related industries. Coupled with the high pressure and tight production timelines in this field, this leads to the unsatisfying situation

that often people working in the gaming industries are not aware of related current research efforts. In a similar way, often little is known in the related research community about how the gaming industry really works and what problems and day-to-day challenges companies involved are facing. Therefore, in addition to the major goal of all STARs – i.e. evaluating the awareness and usage of knowledge technologies in the respective application area – the STAR for Gaming and Simulation is also trying to gain general insight into how the gaming industry works, what common practices and shortcomings are, and if they are in line with current topics addressed by researchers. Together with the other results of the FOCUS K3D project, this STAR should therefore provide a solid basis for the identification of concrete steps to be addressed in related mid and long term projects.

REFERENCES

- More information on FOCUS K3D as well as the AWG Gaming & Simulation can be found at the project's web portal at http://www.focusk3d.eu
- The original STAR reports for all Application Working Groups (including Gaming and Simulation) can be downloaded at http://www.focusk3d.eu/downloads/project-deliverables

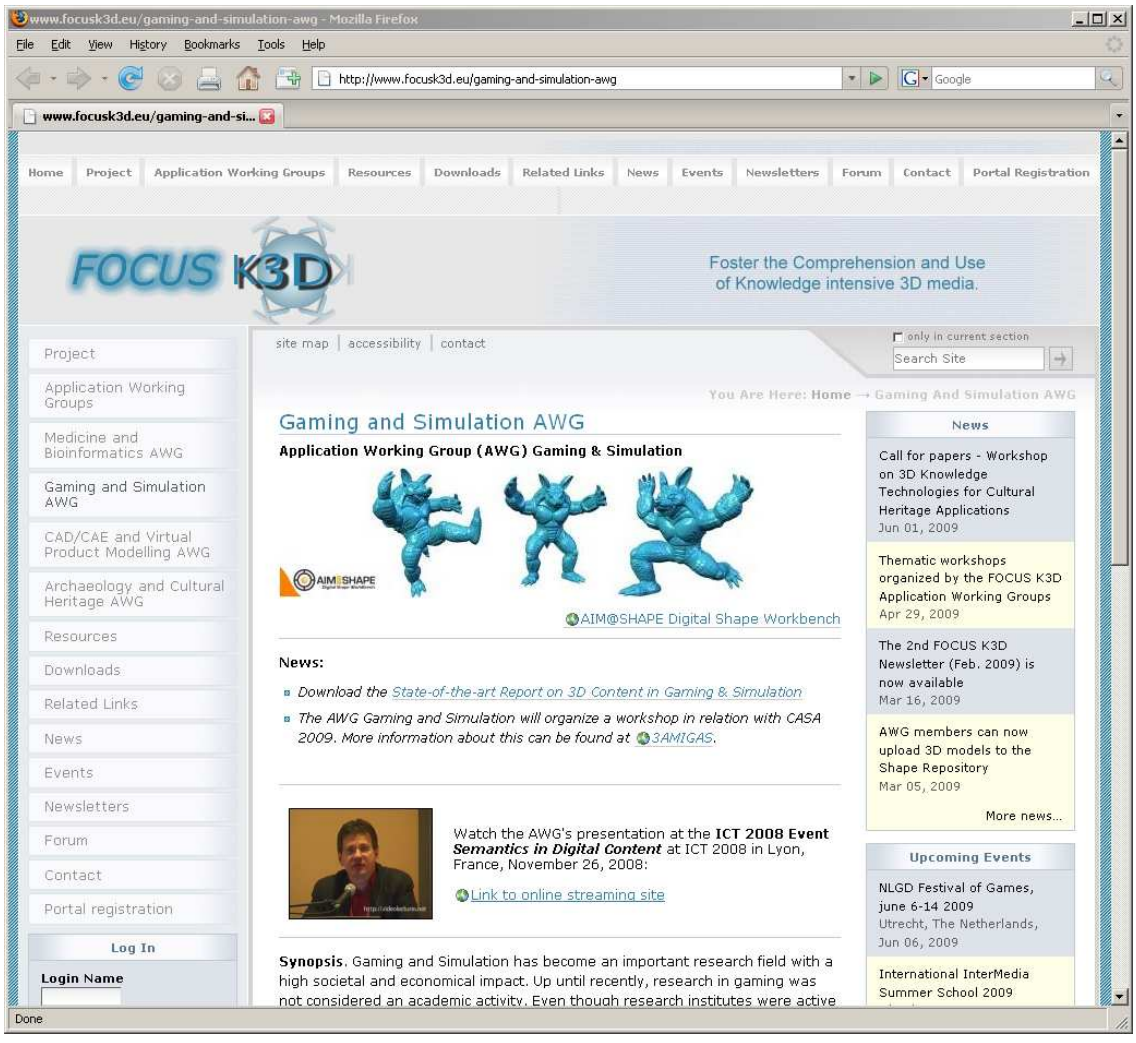


Figure 1.1 The FOCUS K3D web portal, see http://www.focusk3d.eu

2. State-of-the-art Report on 3D Content in Gaming and Simulation

Wolfgang Hürst (Utrecht University, NL), Michela Mortara (IMATI, Italy), Ewald Quak (SINTEF, Norway), Patrick Salamin (EPFL, Switzerland)

In order to evaluate current practices, identify common gaps and problems, and pinpoint directions for future research for modelling and using 3D content, the FOCUS K3D consortium has organized a questionnaire-based survey among several institutions from four different application areas, including Gaming and Simulations. The questionnaires have been assessed and the results formed the basis for a state-of-the-art report (STAR) that describes the current situations among users of 3D content and their knowledge, usage, and expectations of knowledge technologies. The STAR also features an overview of current research topics in this area that have a certain relation to knowledge technologies and an overview of related national and international initiative and projects.

Given the characteristic of the Gaming and Simulation field with small companies, little research, tight production timelines, and a very fragmented branch of industry, it is impossible to draw statistically significant conclusions from such a questionnaire-based survey. However, because the variety of people who participated in this evaluation to some degree mirrors the diversity of this field, the 17 questionnaires that have been considered in the survey provide useful information that enables us to draw reasonable conclusions. The purpose of this document is to summarize these main findings and present them as statements that we put up for discussion.

One of the major motivations for FOCUS K3D is the belief that knowledge technologies can be used in a beneficial way when dealing with 3D data. This seems especially true in the area of Gaming and Simulation, where often not only single objects have to be modelled, but also whole virtual worlds including mutual relations between objects and this virtual world as well as among 3D characters. However, actual evidence, for example, in form of data or established best use cases for this statement can hardly be found. Are knowledge technologies really useful when dealing with 3D data? Are the solutions developed by researchers relevant for actual users and applicable in the production process? We claim that our survey with actual users of 3D and the review done as part of the STAR creation positively answered these questions.

First, our survey showed that although knowledge technologies have not really arrived yet in the practical development, there is a growing awareness of this issue among people working in this area. Although we identified certain misunderstandings and misinterpretations in the used terminology, current users are often well aware of the limitations in the production process for which knowledge technologies could offer a solution. One obvious area for improvement of the current situation is in using knowledge technologies in the actual modelling of 3D characters and objects in order to guarantee that the physical behaviour, characteristics, and restrictions imposed by the virtual world are still met, thus shortening development times and enabling designers to focus on the creative part of modelling instead of having to deal with semantic interpretations of world objects and their relationships. However, our survey also indicated a special characteristic of the area of gaming and simulation, i.e. that the related markets offer high gains and therefore are highly competitive. Hence, it is not only important to provide the best, most realistic models, but also to produce them fast, resulting in a need for an effective and efficient production process. Knowledge

technologies promise to help with this by offering better ways to manage, retrieve, and therefore reuse existing 3D models.

Current research topics in the area of gaming and simulation where we believe that knowledge technologies could play a significant role are modelling the world, virtual characters, simulating the world, interacting with the world, and affective appraisal. Comparing these with the comments made by the participants of the survey indicates that these topics are indeed of high relevance in the actual production process in this area. It appears that both academia as well as industries are aware of the limitations and put high hopes in the improvements knowledge technologies could provide in the future. This growing awareness is also reflected in the review of knowledge technologies that we presented as part of our STAR. Based on these observations, we believe that the application scenarios described will witness a boost in the use of knowledge technologies in the near future.

Although it appears that the topics currently addressed by research seem to be in line with the demands and problems of the industries and practical workers in this field, our survey also revealed a major burden when transferring research results to actual users. Comments made by the participants indicated that research should not only focus on methodologies but also on providing useful tools that fulfil the needs of the actual product designers and developers. In a related issue, the review of existing knowledge technologies revealed a growing consensus on the need of having a listing of concepts that other game designers have found useful.

The growing numbers of national initiatives also show that the rapidly increasing economic importance of games has alerted funding agencies to this subject, focusing mostly on serious games. The (so far not so numerous) EU projects funded in this domain started out with technical issues (like cross-platform integration), often related to eLearning purposes. Only recently semantic issues like ontology development have been incorporated. Given the importance of gaming and simulation and the level of European activities already existing, it seems fair to state that a dedicated set of projects in this area would have a large impact and could yield strong results in serious gaming, involving the major actors in the community, for example through an EU-wide ontology development effort, and in managing 3D content for efficient re-use.

The complete state-of-the-art report can be downloaded for free at the following URL: http://195.251.17.14/downloads/project-deliverables/D2.2.1-final.pdf

3. The AIM@SHAPE Digital Shape Workbench

Chiara Eva Catalano (IMATI, Italy)

The Digital Shape Workbench (DSW) is a common infrastructure intended to be used mainly by Shape Modelling expert users for integrating, combining, adapting, enhancing existing and new software tools and shape databases, developed within the AIM@SHAPE Network of Excellence (see http://dsw.aimatshape.net).

It is composed by separated repositories for different kind of resources, which rely on a conceptualization of the domain achieved through the development of domain ontologies. The purpose of this conceptualization is twofold: on one hand, it offers a unique and shared understanding in the Shape Modelling domain, and on the other hand, it offers the possibility of exploiting reasoning facilities on it.

The DSW infrastructure consists specifically of the data repositories (for shapes and tools, publications), a knowledge management system that handles ontology-driven metadata and a number of different ways of discovering, searching and browsing resources (see Figure 3.1). The basic functional requirements of the DSW are:

- uploading of shape models, tools, bibliographic references and their corresponding metadata according to the ontologies developed by the consortium;
- downloading of resources;
- browsing and searching for resources thanks also to a semantic search engine and a geometric one;
- meta data management activities (e.g. editing, deleting metadata about the resources).

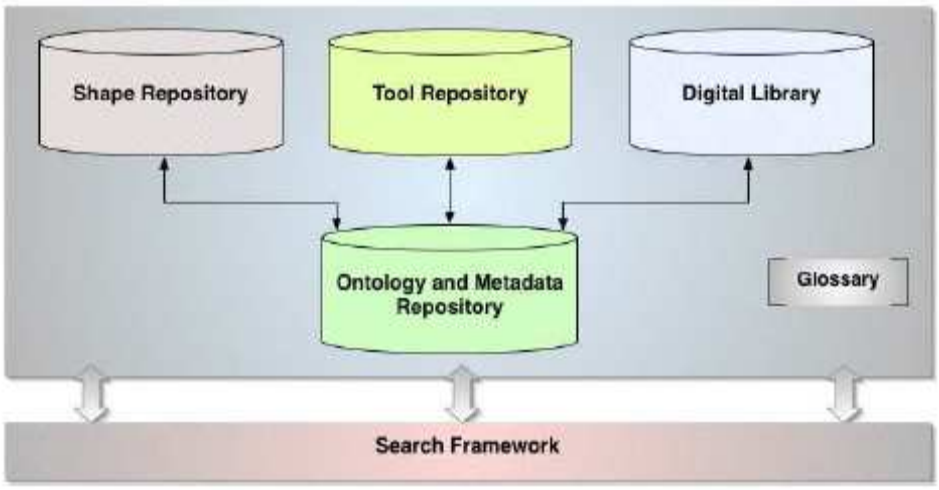


Figure 3.1 The DSW infrastructure.

In particular, the *Shape Repository* is a shared repository populated with a collection of high quality digital shapes in different representations and formats. Its primary goal is to include a variety of standard test cases and benchmarks, enabling efficient prototyping as well as practical evaluation on real-world or large-scale models.

The *Tool Repository* is an inventory of shared software tools that can be used in different stages of digital shape processing. It mainly contains existing tools, developed by the project Partners, but it also combines, adapts, and enhances tools developed by different research teams. In addition, the Tool Repository includes a brief specification of the tools usage, limits and capabilities as metadata.

The *Digital Library* is a common repository of scientific references and technical reports which integrate the bibliographies of the participating Institutes.

The *Ontology and Metadata Repository* constitutes the knowledge base back-end where all domain-specific as well as the more general knowledge, common to the various domains, is stored. The aim is to represent and collect shape knowledge that can be effectively used by real applications and tools. The OMR is an ontology management system which allows storing, editing and accessing ontologies and ontology-driven metadata and is closely related to the semantic search and inference engine.

In AIM@SHAPE two Common Ontologies (COs) have been defined, one capturing shape-related metadata and one related to the tool metadata. They have been integrated and complemented with the development of three ontologies which address the need of particular shape application domains, i.e., Product Design, Virtual Humans, and Shape Acquisition and Processing.

Figure 3.2 The AIM@SHAPE Digital Shape Workbench.

The Shape Repository has been established as a premier source of quality shapes for researchers in geometric modelling, computer graphics and geometry processing. It has received more than 1.5 million visitors and 77,000 downloads since August 2004, and now contains more than 1081 shapes with well-defined metadata. It can be accessed at http://www.focusk3d.eu/resources/digital-shape-workbench (cf. Figure 3.2). We invite researchers and partners from the AWGs and beyond to contribute to the shape repository by uploading new models coming from the four application domains covered by the FOCUS K3D project, and Gaming and Simulation in particular, contacting the AWG leader.

4. Semantic Processing of a 3D Game Character Model

Wolfgang Hürst (Utrecht University, NL), Michela Mortara (IMATI, Italy), Chiara Eva Catalano (IMATI, Italy), Marco Attene (IMATI, Italy)

Despite an obvious need for related functionalities, knowledge technologies are commonly not used in the gaming industries [1]. The reasons for this are unclear. Are related research results too new and not ready for practice yet? Are they not relevant for the problems people in the gaming industries are actually facing? One of the main objectives of the Coordination Action FOCUS K3D (cf. Section 1) is to answer such questions in order to promote the adaption of semantics and knowledge technologies in 3D content modelling and processing in four application areas including Gaming & Simulation.

Unfortunately, transfer of knowledge and research results from academia to the gaming industry does not always work very well. Also, research prototypes are often demonstrated using mock-up data that is different from the actual models used in the industries. In the following, we summarize how we used current, state-of-the-art knowledge technologies and semantic processing tools [Glossary: *semantic*] that have developed at the Institute for Applied Mathematics and Information Technologies (IMATI) in Genoa, Italy, on a 3D model from a game that was provided by Silent Bay Studios s.n.c., an independent game developer company [2]. We will analyse the model, segment the shape into significant parts by means of a tool called Plumber, annotate each part with a concept by means of the ShapeAnnotator tool and finally envisage a search of similar models or of similar parts for modelling purposes. The purpose of this is to demonstrate with an actual example from a real game how current, state-of-the-art technologies could be used in a beneficial way in game development, and also to identify obvious problems and issues that should be addressed in order to guarantee a better transfer of the promising research results into the actual development and design process. It should be noted, that the related statements reflect the personal opinion of the authors. Obviously, we can not draw general conclusions from processing just one single model. Nevertheless, we believe that the presented observations provide valuable insight that can help in creating a better match between the actual needs of the gaming industries and the research done by academia.

The Model – Introducing Gurk

It appears to us, that in a gaming context, the most important aspect often seems to be the appearance of the model rather than its geometry. For example, Gurk (cf. Figure 4.1, left) is a game character provided to the FOCUS K3D project by Silent Bay Studios s.n.c. For rendering purposes, the Gurk model exhibits highly complex geometry: it is non manifold and it is constituted by more than 120 connected components (shells). Oppositely, the shape analysis tools developed at IMATI that we want to apply in this example are designed to handle single-component, manifold triangle meshes [Glossary: *mesh*, *manifold*]. As a consequence, some pre-processing with the original model had to be done first.

Tools: **ReMESH** [T1] is a powerful editor for manifold triangle meshes with advanced repairing features

Pre-processing – Simplification of the model

In order to simplify the complexity of the model and make it easier to handle for our segmentation tools, we first processed it with the **Remesh** tool to obtain a single shell and a manifold surface. This took quite a bit of work including low level manual operations; we also had to remove some accessory components

(like the flag) and some detail features (like the top of the head protrusions) to obtain a manifold representation. After such operations the size of the model increased; therefore we also applied a simplification step. The result of this pre-processing is illustrated in Figure 4.1. Table 4.1 summarizes the concrete data.

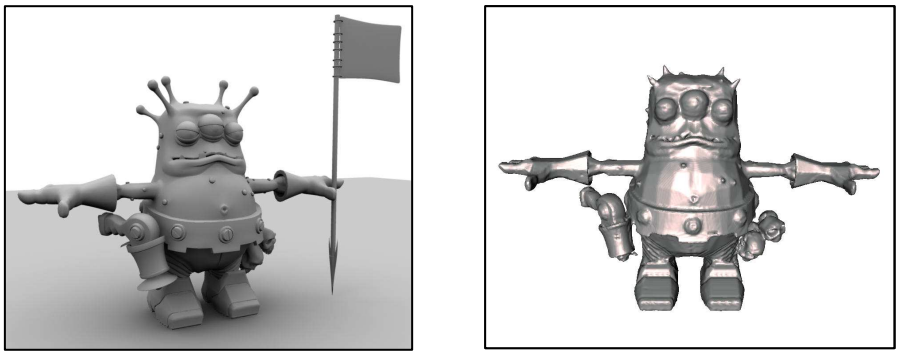


Figure 4.1 Pre-processing of the original Gurk model (left) leads to a less complex version that can be handled by common geometry processing tools (right).

Original model		Manifold model		Simplified model	
Vertices	46670	Vertices	195731	Vertices	19969
Edges	136967	Edges	587211	Edges	59925
Triangles	90374	Triangles	391474	Triangles	39950
Boundaries	121	Boundaries	0	Boundaries	0
Holes	28	Holes	4	Holes	4
Shells	127	Shells	1	Shells	1

Table 4.1 Detailed data of the models before and after pre-processing.

Segmentation – Putting Gurk into pieces

On the manifold simplified model we were able to run the **Plumber** segmentation [Glossary: *segment, segmentation*] and identify several tubular components with their axis that can be a support to the creation of the skeleton and can provide partial skinning data to animate Gurk (cf. Figure 4.2).

Tools:
Plumber [T2] is an interactive interface for multi-scale segmentation of triangulated surfaces

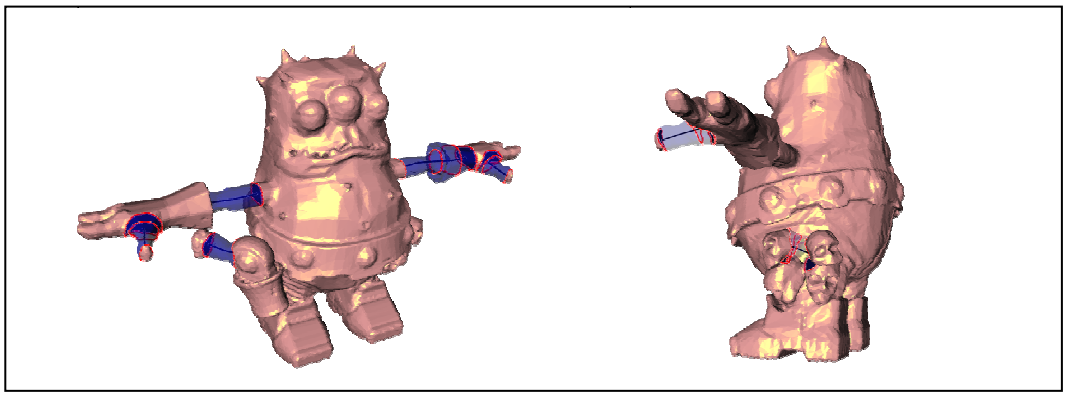


Figure 4.2 Identification of some tubular components of Gurk by Plumber.

Another benefit of the segmentation produced by Plumber is that we can use it to calculate a "best view" of Gurk, i.e. a position that gives the most informative image of the model in terms of its relevant features (identified by the segments)

[Glossary: *view*]. This application is very useful for instance for generating thumbnails for large data sets of models (like characters and objects already modelled that could be used to create a new virtual world). The result is illustrated in Figure 4.3.

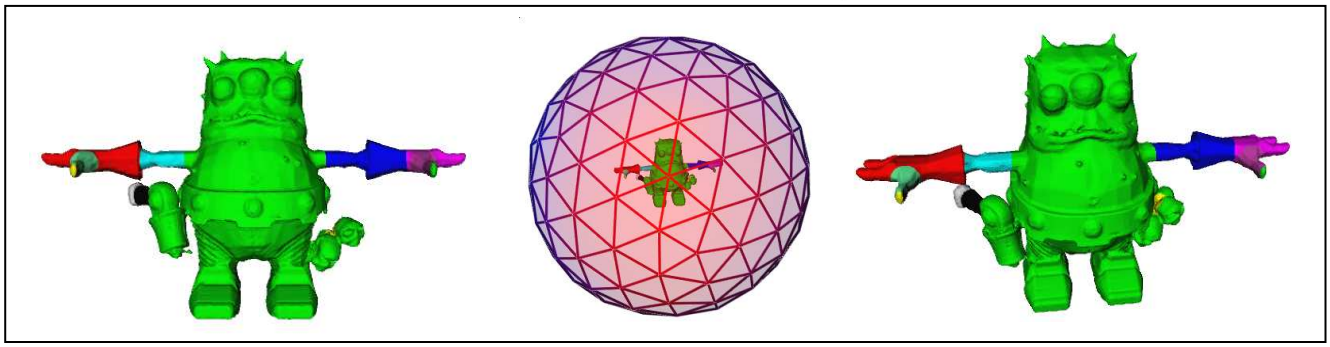


Figure 4.3 Getting the "best view" of Gurk. Left: the Plumber segmentation. Centre: Sampling of the admissible viewpoints on a surrounding sphere and colouring according to the "goodness" of the corresponding image. Right: the computed best view that maximizes the percentage of visible segments.

Shape annotation – Describing Gurk and its pieces

Tools: The **Shape-Annotator** [T3] provides an interactive GUI for the inspection, segmentation and semantic annotation of 3D surface meshes

The **ShapeAnnotator** was applied to annotate the body parts of Gurk on the base of a simple ontology that formalizes the human body structure [Glossary: *ontology*]. Using different segmentation techniques (Reeb graph and Fitting Primitives) we achieved a multi-segmentation of the model (depicted in the left window of Figure 4.4 and 4.5 in the following) and we selected the segments that in our opinion match the ontology concepts (head, torso, arms...). Once selected, a segment appears in the right-hand window. Then we refined the segmentation in the right window by manually editing segments (the ShapeAnnotator allows to create a new segment, merge two adjacent segments, shrink/grow, and flood a segment). The final segmentation obtained in the right window is ready to be annotated with concept of the ontology.

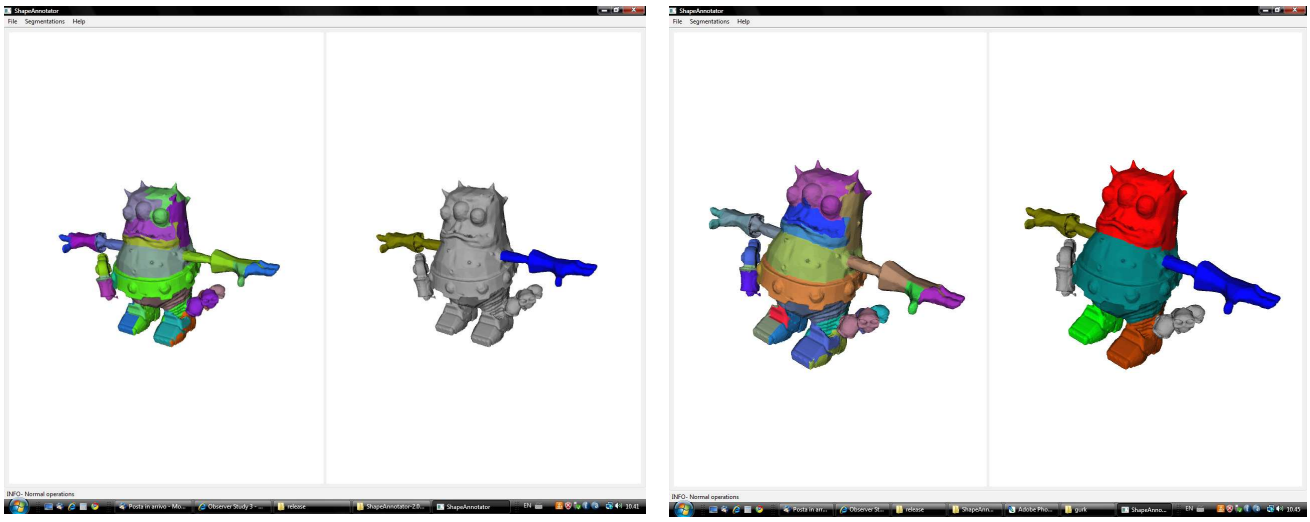


Figure 4.4 Two steps in the selection of segments from those automatically computed by the system. The right window shows the segments selected by the user.

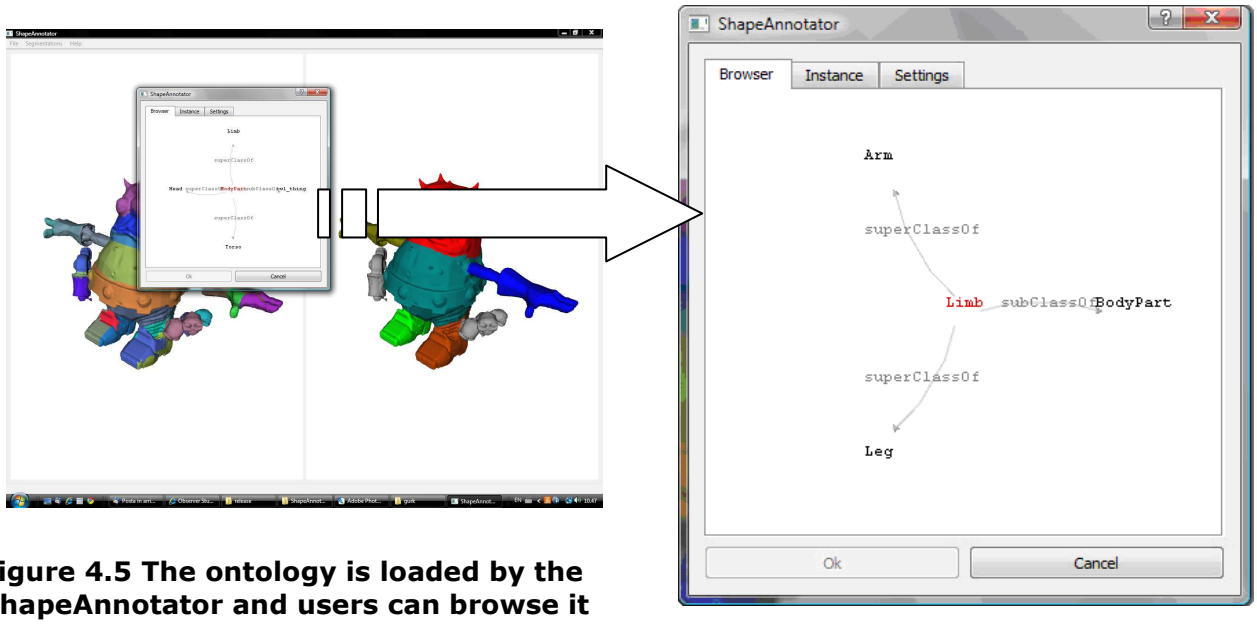


Figure 4.5 The ontology is loaded by the ShapeAnnotator and users can browse it to annotate each desired concept.

When a segment is annotated, an instance is created in the ontology which encodes the segment parameters as instance attributes and relations, and also maintains the link to the segment geometry as a dedicated URI. Moreover, some geometric/topological attributes of each instance can be automatically computed by the system on demand (e.g. arm length, body part adjacency, cf. Figure 4.6) thanks to the *Segmentmeters,* which work on the geometry of the objects, independently of any ontology, and they can be interpreted by the user according to the specific domain of annotation.

All the resources are encoded as an owl file as illustrated by the following extract:

```
<?xml version="1.0"?>
<rdf:RDF
    xml:base="http://www.ge.imati.cnr.it/virtualHuman.owl"
    xmlns="http://www.ge.imati.cnr.it/virtualHuman.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#">
  <owl:Ontology rdf:about="">
  <owl:imports
rdf:resource="file:/E:/My_Software/ShapeAnnotator/ShapeAnnotator-2.0-
1alpha_src/Ontology/default_ontology.owl"/>
    <rdfs:comment xml:lang="en">Created by ShapeAnnotator
2.0beta</rdfs:comment>
  </owl:Ontology>
<owl:DatatypeProperty rdf:ID="Shann_Segment_ID">
<rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
<rdfs:domain>
```

```
<owl:Class>
<owl:unionOf rdf:parseType="Collection">
<owl:Class rdf:about="#Torso"/>
<owl:Class rdf:about="#Head"/>
<owl:Class rdf:about="#Arm"/>
<owl:Class rdf:about="#Leg"/>
...
<Arm rdf:ID="Arm_1243586910">
  <Adjacent_To rdf:resource="#Torso_1243586898"/>
  <length
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">59.695655</length>
  <Shann_GeoContext_URI
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">file:/C:/Users/
jaiko/Desktop/gurk/gurk.ply</Shann_GeoContext_URI>
  <Shann_Segment_ID
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">3</Shann_Segment_I
D>
</Arm>
```

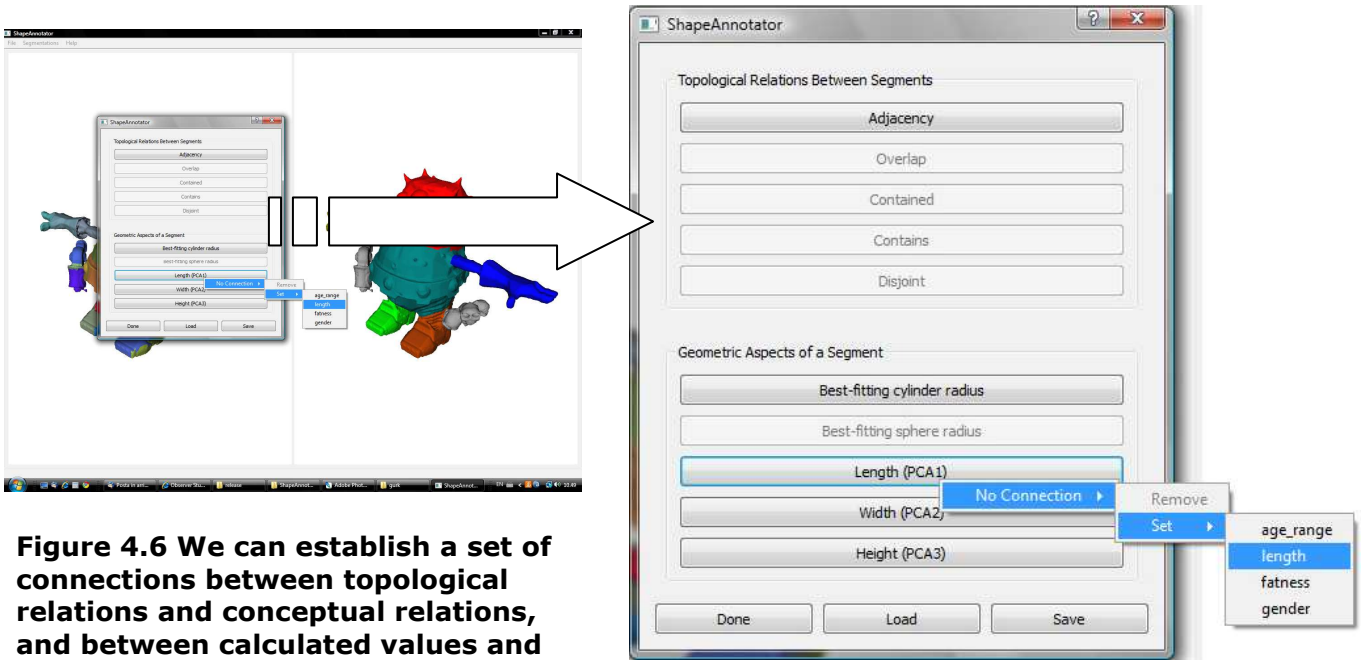


Figure 4.6 We can establish a set of connections between topological relations and conceptual relations, and between calculated values and class attributes (arm length in this example).

For the above annotation, a simple ontology describing the human body parts was used. It should be noted, that using this basic ontology the gun and the little skull features as well as the small "horns" on the head could not be annotated since no concept is available for that features. Indeed it would be much more interesting to develop a more general ontology for game characters which takes into account many features and accessories that a fantasy creature may exhibit.

After annotating character models, an animator could interrogate the resulting knowledge base to retrieve by means of a concept-driven query not only a whole model, but also specific body parts. For example one could retrieve different legs, arms, and heads to compose a new model by re-using parts of pre-existing models.

Geometric and semantic searches – Finding Gurk (and his look-alikes)

The Gurk model used in the semantic processing steps above was provided by Silent Bay Studios s.n.c. and is available in the **Digital Shape Workbench** (DSW, [3]). The DSW was developed as part of AIM@SHAPE – a Network of Excellence project within the European Union's 6th Framework Programme (cf. section 3). Conversely to the ShapeAnnotator that is targeted at the part-based annotation of models, the DSW annotates whole models according to geometric/structural and semantic properties. Thanks to the semantic and geometric search facilities, models similar to Gurk from the geometric/structural or conceptual point of view can be now retrieved.

Tools: **DSW, the Digital Shape Workbench** [T4] is a common infrastructure for integrating, combining, adapting, and enhancing existing and new software tools and shape databases

Moreover, Gurk could be included as an instance of the Virtual Human Ontology [Glossary: *virtual human*], where he can be also annotated with semantic information such as his animation sequence, structural description and landmarks extracted, and his emotional state, if any. In this way, Gurk could be retrieved from the repository also thanks to his context-specific characteristics.

CONCLUSION

The examples presented above illustrate that the combination of 3D graphics with current knowledge technologies and approaches for semantic processing could have great benefits in game design – even if they have not been developed for this particular application. Tasks such as support for skeleton creation, automatic thumbnail generation of best views, annotation support, automatic computation of attributes such as arm length, and improved re-usage and modelling by concept-driven retrieval seem to be obvious examples for potential improvements in the game development process.

However, we also identified obvious problems and difficulties. First, the Gurk model needed great efforts in geometric pre-processing to be treated by our common geometry processing tools. This could indicate a need for research in computer graphics to meet related conversion issues or to adapt their analysis tools to work on more generic models. Due to no need for manifold surface and the large variety of shapes, the necessity to interface them with geometry processing tools that might be suited only for specific presentation types seems to be a critical issue. In addition, a significant part of the semantic annotation cannot be extracted geometrically and thus has to be manually included. Finally, a general "gaming creature ontology" does not exist (yet), but would be very helpful for annotation, retrieval, and re-usage purposes.

Acknowledgements

The authors like to thank Silent Bay Studios for providing the Gurk model used in these experiments and their valuable input, as well as Patrick Salamin from the VRlab at EPFL in Switzerland and the other FOCUS K3D partners and members who provided valuable feedback and support.

References

[1] W. Hürst et al. (2009) State-of-the-art Report on 3D Content in Gaming & Simulation, available at http://www.focusk3d.eu/downloads/project-deliverables/D2.2.1-final.pdf

[2] Silent Bay Studios s.n.c., see http://www.silentbaystudios.com

[3] The Gurk model in the AIM@SHAPE Digital Shape Workbench, see http://shapes.aimatshape.net/view.php?id=1471

Tools

[T1] ReMESH – Edit and Repair Polygon Meshes, see http://remesh.sourceforge.net/

[T2] Multi-scale Shape Segmentation, see http://www.ima.ge.cnr.it/ima/smg/plumber-web/plumber-web.html

[T3] ShapeAnnotator – Part-based Annotation of 3D Surface Meshes, see http://shapeannotator.sourceforge.net/

[T4] Digital Shape Workbench, see http://dsw.aimatshape.net/

Glossary

Excerpt from the AIM@SHAPE DSW Glossary. For more related terms see AIM@SHAPE DSW glossary at http://dsw.aimatshape.net/glossary/

Manifold – A (separable Hausdoff) k-dimensional topological space M in which each point has a neighbourhood which is homeomorphic either to the k-dimensional open ball, or to the half-ball.

Mesh – A grid-like polygonal subdivision of the surface of a geometric model.

Ontology – The specification of a conceptualisation of a knowledge domain. In philosophy (from the Greek "on" = being/to be and "logos" = word/speech) it is the science or study of being or existence as well as the basic categories thereof - trying to find out what entities and what types of entities exist. In computer science, an ontology is the product of an attempt to formulate an exhaustive and rigorous conceptual schema about a domain. An ontology is typically a hierarchical data structure containing all the relevant entities and their relationships and rules within that domain (e.g. a domain ontology). On the Web, the term applies to the many ongoing efforts to develop topic-specific sets of XML-friendly language, rules and definitions. Essentially, IT ontologies are logical supersets that combine metadata, taxonomies and semantics into formal systems that can be encoded in software, enabling diverse Web applications to truly understand one another.

Segment – (1) Portion of an H-Anim figure (Virtual Human) or Smart Object that represents its geometry and appearance. (2) A collection of pixels that have similar properties (e.g., similar intensity values or texture) which differ from their surrounding pixels. (3) A subset of a data set classified according to similar characteristics of the basic entities or to given criteria.

Segmentation – To split a data set into smaller subsets based on given classifications or criteria. Frequently used in image processing to analyse pictures and for detection. When reconstructing geometry objects from sampled points, segmentation techniques can be used to detect subsets of points belonging to the same mathematical surface.

Semantic – Pertaining to meaning (from the Greek "semainein" = to signify/mean). Recently there has been much talk about the emergence of the "Semantic Web".

Semantic Web – Framework in which applications and information are organized in such a way that applications are intelligent systems where computers can effectively "understand" the meaning of the information, unlike HTML-based systems that are mostly concerned with how information is displayed.

View – In a 3D modeller, a view indicates a projection of an object according to some planes, typically the main planes of the coordinate reference system. In the

product development process, a view is the product information required by a development phase. Since the various phases of the development process have different goals, they require proper digital model and information of the product itself together with proper tools to treat them. It is therefore necessary to transform the model from one point of view to another one.

Virtual Human – Specialized instance of an articulated character. The model can be synthesised in a variety of ways and can represent a real or a virtual person. VHs are characterized by a set of general attributes (sex, nationality, race, ...), and structural descriptors (skeleton, geometry, landmarks, etc.).

5. Position Statements

Wolfgang Hürst (Utrecht University, NL), Michela Mortara (IMATI, Italy), Ewald Quak (SINTEF, Norway), Patrick Salamin (EPFL, Switzerland)

The questionnaire-based survey, literature study, and review of the Gaming and Simulation domain that was done as part of the STAR writing by the FOCUS K3D AWG Gaming and Simulation (cf. section 2) and the experience gained from applying our processing tools to a 3D model from an actual game (cf. section 4) provided us with some valuable insight about if and how 3D modelling in the gaming domain could benefit from the use of semantics. Obviously, some of our conclusions might be controversial due to the limited number of samples that were available for the STAR and the restriction to one model and our own processing tools in the concrete use case. Therefore, we like to put our conclusions up for discussion and invite everyone to join the debate. In particular, we formulize several observations as statements for which we are asking for you feedback.

Perspectives and opportunities: Based on the observations in the STAR, we conclude that gaming and simulation is a very active, highly competitive field with increasing economic importance. This results in the somehow contradicting requirements of producing high quality, state-of-the-art products on the one hand vs. the need to provide them fast and therefore constantly improve the production process and make it more effective. We claim that the use of semantic processing tools could help in this context, in particular in the following two aspects:

- There is a strong need for a better, uniform, understandable, and re-usable way to create and process 3D models (e.g. tools with moving parts controlled by input parameters, objects that react differently in varying temperatures, crowds of people or animal herds where subjects look and behave similar but are not identical). Although not providing a completely satisfying solution yet, knowledge technologies appear to be a promising step into the right direction to satisfy this need (e.g. by providing segmentation tools that enable you to easily identify and re-use parts of an existing model, ontologies that guarantee a certain behaviour when a model is modified).
- Reuse, version control, and meta data management are becoming more and more important especially with larger projects and more complex models but the same or even shorter production cycles. Tools that allow you, for example, to search for similar shapes such as the DSW described in section 3 or ambitions to create a Google-like web-based search engine for 3D models that supports meta data such as polygon-count, texture information, etc. as well as information on which project it was used, what people created and worked on it, etc. can therefore provide high value for the gaming industry.

Problems and critical issues: Our discussion with different people working in the gaming and simulation domain as well as the experience from processing an actual model with our tools did not only demonstrate tremendous opportunities for the beneficial usage of semantics in 3D modelling, but also highlighted several problems. First, the interviews and discussions done as part of the STAR research revealed an often existing apparent misunderstanding and miscommunication among members form academia and the industries, leading to the following claim:

- Academia needs to do a better job in promoting knowledge technologies and the related terminology. In addition, the actual users need to better specify their real needs by more accurately describing their workflows and production processes.

In addition to this high-level statement, the Gurk example (cf. section 4) revealed some concrete problems and limitations that have to be addressed in order to exploit the full benefit of using semantics for 3D modelling in the gaming domain:

- It appears to us that there is a strong need for conversion tools in order to deal with a variety of models on the one hand and geometry processing tools that have only been designed for specific presentation types on the other hand.
- Although promising, many tools for semantic annotation still require a non-significant amount of manual labour; an issue that seems especially critical in the gaming domain and therefore should be addressed by research.
- Ontologies are an important part of many approaches for semantic processing but ones that are suitable for gaming related characters and objects are still rare or limited. Thus, gaming ontologies formalising the different aspects, for example, of the virtual creature pipeline, such as creation, animation, interaction, etc. are necessary.

Given the diversity of the field, we expect that not everyone agrees with our conclusions and several people might have different experience and other opinions. We encourage everyone to share them with us and by doing so, to contribute to the ultimate goal of FOCUS K3D which is to foster the comprehension, adoption and use of knowledge intensive technologies for coding and sharing 3D media content in consolidated and emerging application communities such as gaming and simulation. We hope that this discussion results in propositions for concrete examples for immediate steps to be taken in new research activities that address specific aspects of the identified challenges and we encourage everyone to join these efforts.