Defining spatial relations in a specific ontology for automated scene creation

Contraş Diana, Pintescu Alina
Technical University of Cluj-Napoca
Baia Mare, Romania
pdia17@yahoo.com, taly_74@yahoo.com

Abstract— **This paper presents the approach of building an ontology for automatic scene generation. Every scene contains various elements (backgrounds, characters, objects) which are spatially interrelated. The article focuses on these spatial and temporal relationships of the elements constituting a scene.**

Keywords— class, individual, object property, ontology, relation, scene

I. INTRODUCTION

The scenes are ubiquitous. At any time, in any place we encounter scenes: in plays, in pictures, in films, in everyday life. All scenes contain various items such as backgrounds, characters, objects. To get a consistent scene elements must be related by spatial relationships. If we want the perfect scene for us, meaning a scene to include the desired items, it could be difficult to create it manually. An automatic system, specialized in this regard, would be useful. Using it we could choose the favorite items to generate the wanted scene. If the result is not satisfactory may return the choice made and generate again. Repeating the process we could get the perfect scene.

A term from artificial intelligence (AI) refers to ontology. In [8] Gruber gave the most simple definition for the ontology - an explicit specification of a conceptualization. The elements of scene and the spatial relationships between them can be represented using ontologies.

In this article we focus on the spatial relationships of a precisely ontology in order to mark out the importance of ontology in the process of automatic generation of the scenes.

II. RELATED WORK

The term *scene* may refer to arts and media, music and culture, science and technology and many other domanins. We find a scene in a film, we see a scene in a picture, we identify a scene as a moment in a restaurant or hospital or school. Could be quite difficult to shoot a film or to paint a picture or to describe an event that happened at a time. Our intention is to automate the generation of scenes regardless of the field of application.

In recent years we find in the literature software specialized in generating images.

In [2] is presented a methodology for the automated orientation of image blocks acquired with calibrated cameras and the successive object 3D reconstruction. All is done using two software: *ATiPE* and *CLORAMA*.

CCP4mg [15] is software created in order to represent macromolecular structures. It has a wizard that facilitates the generation of complex scenes.

The Painting Fool is a software that generates automatically the paintings. His author, Simon Colton of Imperial College, London, hopes that one day his product will be taken seriously as a creative artist [4].

CarSim is a software created for automatic text-to-scene conversion. It analyzes written descriptions of car accidents and creates 3D scenes of them [1].

The WordsEye system is another text-to-scene software generation. With this software users can create 3D scenes only through natural language without special skill or training [6].

OntoPlant is a software package developed at Spatial Information Research Center of Fujian in Fuzhou University since 2002. The purpose of the software package is to provide an integrated software solution to realistic plant modelling, real-time scene rendering, growth simulation and applications at different scales from individual, stand (population, community) to landscape [18].

In [19] is presented a system that automatically generates multiple furniture arrangements based on hierarchical and spatial relationships between objects.

In [11] is presented not a scene generation, but an image registration process. We mention it because are implemented two genetic algorithms with two selection criteria and both of them are proved to be feasible for image registration.

Another usage of genetic algorithms is presented in [3] to evolve vector images which are composed of lists of discrete geometric shapes, such as circles, squares, and lines and are popular in illustration and graphic design.

Ontologies are used in many areas from technological processes [10], [16], [17] to medical field [12], [14].

From a technical standpoint, the combination of ontologies with other artificial intelligence methods like genetic algorithms [13] or multi-agent systems [5] result in superior innovative solutions.

Novelty we bring is the representation of scene elements and relationships between them using ontologies. In order to generate scenes automatically we will use ontology as population of genetic algorithms. In this article we insist on presenting relations between entities present in such an ontology.

III. Structuring the Ontology

A. Classes

In [14] is mentioned the existence of more than 50 ontology editors. Matei enumerate the reasons for choosing the editor Protégé, a free-open source from Stanford University. An ontology built in Protégé consists of Classes, Slots and Instances [9].

Before we present the relationships between classes we have to list the class hierarchy. In everyday life any scene consists of **Frame** and **Content**. These two will be the supercalsses of the ontology. A frame may refer to **Interior** or **Exterior** environment. The content has been adapted to these two situations: **ExteriorContent** and **InteriorContent**. Further classes are customized by introducing appropriate subclasses.The Fig. 1 illustrates the tree structure of the ontology.

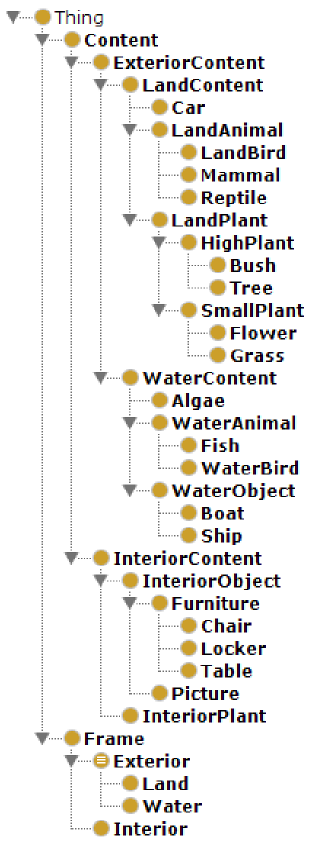


Fig. 1. Class hierarchy

B. Object properties

An ontology is consistent only when establishing relationships between entities. In an ontology object properties ensure the relationship between two individuals [7]. It could be the same individual or different individuals.

Each scene consists of a frame and content. To build the scene must place items that represent the content in a certain logical order. For this reason the relationship between elements should express an element positioning in space by all or some of the other elements that make up the scene. When we place an object in space we use prepositions that establish the precise place of it depending of other objects: above, below, in front of, behind, left, right.

Based on mereology's axioms and primitive notions and mathematical notations we define the object properties for this ontology. R(x, y) is the convention for individual x is related to individual y via relationship R.

$$\text{hasAbove}(x, y)=\{\exists x\in \text{Furniture}\ \exists y\in \text{Picture}\} \qquad (1)$$

hasAbove - the domain is **Furniture** and the range is **Picture**; we consider that the furniture can have above pictures.

$$\text{hasBehind}(x, y)=\{\exists x\in \text{HighPlant}\ \exists y\in \text{LandContent}\} \qquad (2)$$

hasBehind - the domain is **HighPlant** and the range is **LandContent**; in a scene that contains tall plants like bushes and trees can be placed behind them a number of items such as cars or animals.

$$\text{hasBelow}(x, y)=\{\exists x\in \text{WaterObject}\ \exists y\in \text{WaterContent}\} \qquad (3)$$

hasBelow - the domain is **WaterObject** and the range is **WaterContent**; ships and boats float on water and below them may be algae, fish and aquatic birds.

$$\text{hasInFrontOf}(x, y)=\{\exists x\in \text{HighPlant}\ \exists y\in \text{SmallPlant}\} \qquad (4)$$

hasInFrontOf - the domain is **HighPlant** and the range is **SmallPlant**; in a scene we want to place small plants in front of high plants to be visible.

$$\text{hasInterior}(x, y)=\{\ \exists x\in \text{Interior}\ \exists y\in \text{InteriorContent}\} \qquad (5)$$

hasInterior - the domain is **Interior** and the range is **InteriorContent**; is a more general relation which places all interior things in their appropriate medium.

$$\text{hasLand}(x, y)=\{\ \exists x\in \text{Land}\ \exists y\in \text{LandContent}\} \qquad (6)$$

hasLand - the domain is **Land** and the range is **LandContent**; is the same type of relation as the above one, but it refers to land medium.

$$\text{hasLeft}(x, y)=\{\exists x\in \text{Content } \exists y\in \text{Content}\} \quad (7)$$

hasLeft - the domain is **Content** and the range is Content; is a relation that help us to place an object related to another object which is already placed on scene.

$$\text{hasRight}(x, y)=\{\exists x\in \text{Land } \exists y\in \text{Water}\} \quad (8)$$

hasRight - the domain is **Land** and the range is **Water**; if we would want to insert in a picture land and water we would have to know how to arrange them.

$$\text{hasWater}(x, y)=\{\exists x\in \text{Water } \exists y\in \text{WaterContent}\} \quad (9)$$

hasWater - the domain is **Water** and the range is **WaterContent**; is the third general relation which places the water content on water.

Each object property may have inverse property [9]. The inverse of a binary relation R(x, y) is the binary relation R^{-1} defined: $R^{-1}(x, y)$ if and only if R(y, x). It means that if the property links individual **y** to individual **x** then the inverse property links individual **x** to individual **y**. In this ontology every object property has his inverse propety:

$$\text{isAbove}(y, x)=\textit{has Above}^{-1}(x, y) \quad (10)$$

hasAbove - has the inverse property **isAbove**; if the furniture has above it a picture, then we can say that the picture is above furniture.

$$\text{isBehind}(y, x)=\textit{hasBehind}^{-1}(x, y) \quad (11)$$

hasBehind - has the inverse property **isBehind**; if a bush or a tree has behind it a car or an animal, then that car or animal is behind the heigh plant.

$$\text{isBelow}(y, x)=\textit{hasBelow}^{-1}(x, y) \quad (12)$$

hasBelow - has the inverse property **isBelow**; if a ship or a boat has below it algae, fishes or aquatic birds, then those algae, fishes or aquatic birds are below the ship or boat.

$$\text{isInFrontOf}(y, x)=\textit{hasInFrontOf}^{-1}(x, y) \quad (13)$$

hasInFrontOf - has the inverse property **isInFrontOf**; if a bush or a tree has in front of it a small plants like grass or flowers, then the small plants are in front of high plants.

$$\text{isInterior}(y, x)=\textit{hasInterior}^{-1}(x, y) \quad (14)$$

hasInterior - has the inverse property **isInterior**; if an interior frame contains interior objects, then those interior objects are contained in that interior frame.

$$\text{isLand}(y, x)=\textit{hasLand}^{-1}(x, y) \quad (15)$$

hasLand - has the inverse property **isLand**; if an exterior frame like land contains land objects, then those land objects are contained in that specific frame.

$$\text{isLeft}(y, x)=\textit{hasLeft}^{-1}(x, y) \quad (16)$$

hasLeft - has the inverse property **isLeft**; if an object O1 has in its left side another object O2, then O2 is on the left side of O1.

$$\text{isRight}(y, x)=\textit{hasRight}^{-1}(x, y) \quad (17)$$

hasRight - has the inverse property **isRight**; if a piece of land has water in its right side, then the water is on the right side of the land.

$$\text{isWater}(y, x)=\textit{hasWater}^{-1}(x, y) \quad (18)$$

hasWater - has the inverse property **isWater**; if an exterior frame like water contains water objects, then those water objects are contained in that specific frame.

For a more precise positioning of content into the framework we build particular spatial relationships by composing the existing ones.

$$\text{hasLeftAbove}(x, y) = \text{hasLeft}(x, y) \circ \text{hasAbove}(x, y) = \{\exists x\in \text{Locker } \exists y\in \text{Picture}\} \quad (19)$$

The relationship **hasLeftAbove** is the result of the composition of two existing relationships **has Above** and **hasLeft**. It was created to toggle a picture not only above a locker, but at the left side of it.

$$\text{hasLeftBehind}(x, y)=\text{hasLeft}(x, y) \circ \text{hasBehind}(x, y) = \{\exists x\in \text{Tree } \exists y\in \text{Car}\} \quad (20)$$

hasLeftBehind is a relationship created by the compositon of the relationships **hasLeft** and **hasBehind**. It puts a car behind the left side of a tree.

$$\text{hasLeftBelow}(x, y)=\text{hasLeft}(x, y) \circ \text{hasBelow}(x, y) = \{\exists x\in \text{Ship } \exists y\in \text{Fish}\} \quad (21)$$

The composition of **hasLeft** relationship to **hasBelow** relationship generates **hasLeftBelow** relationship, which always places the fishes not only under the ships, but in the left side of them.

$$\text{hasLeftInFrontOf}(x, y)=\text{hasLeft}(x, y) \circ \text{hasInFrontOf}(x, y) = \{\exists x\in \text{Bush } \exists y\in \text{Flower}\} \quad (22)$$

The relationship **hasLeft** places an object on the left of another object. The relationship **hasInFrontOf** places the small plants in front of high plants. The composition of these two relations results in a relationship that indicates the positioning of flowers in front, to the left of the bushes.

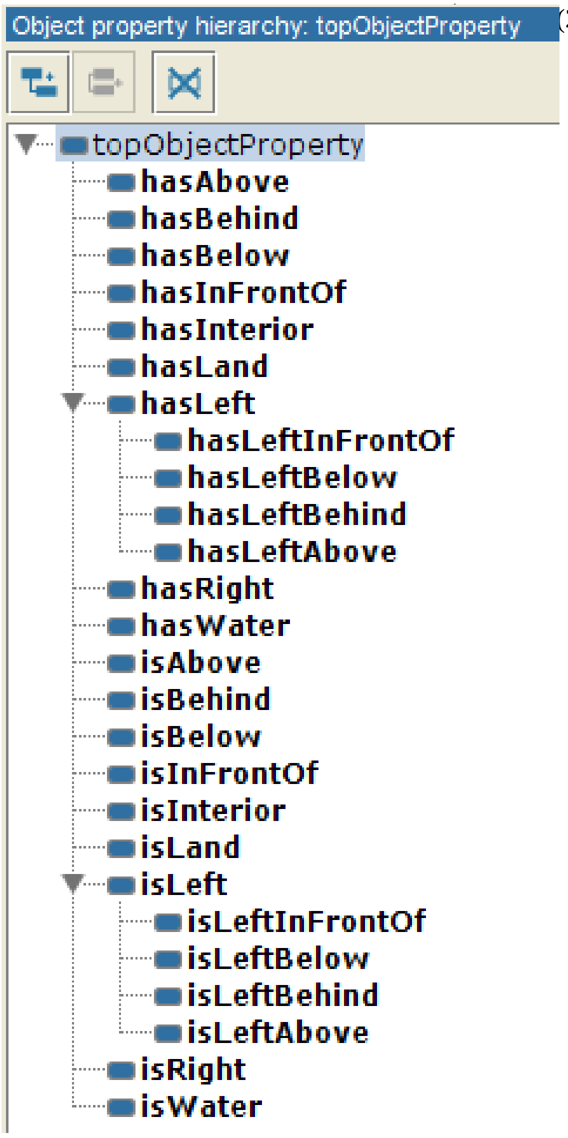
The last four relationships have inverse properties:

$$\text{isLeftAbove}(y, x)=\textit{hasLeftAbove}^{-1}(x, y) \quad (23)$$

$$\text{isLeftBehind}(y, x)=\textit{hasLeftBehind}^{-1}(x, y) \quad (24)$$

$$\text{isLeftBelow}(y, x)=\textit{hasLeftBelow}^{-1}(x, y) \quad (25)$$

(26)



Object property hierarchy: topObjectProperty

```
topObjectProperty
    hasAbove
    hasBehind
    hasBelow
    hasInFrontOf
    hasInterior
    hasLand
    hasLeft
        hasLeftInFrontOf
        hasLeftBelow
        hasLeftBehind
        hasLeftAbove
    hasRight
    hasWater
    isAbove
    isBehind
    isBelow
    isInFrontOf
    isInterior
    isLand
    isLeft
        isLeftInFrontOf
        isLeftBelow
        isLeftBehind
        isLeftAbove
    isRight
    isWater
```

Fig. 2. Object property hierarchy

An object property between two classes **A** and B is *functional* if ∀ a∈A ∃ at most one individual b∈B that is related to the individual a through the property.

$$\forall\ x\in\text{Land}\ \exists\ \text{at most one individual}\ y\in\text{Water} \quad (27)$$

We consider that in a scene that includes both land and water is sufficient to present a piece of each. Therefore the relationship **hasRight** is functional.

If ∀ x R(x, x) the relationship is *reflexive*; if ∀ x ¬R(x,x) the relationship is *irreflexive*. An object property is *irreflexive* when an object is never in the relation to itself.

$$\forall x\in\text{HighPlant}\ \neg\text{hasBehind}(x, x) \quad (28)$$

The object property **hasBehind** relates an individual from the class **HighPlant** with an individual from the class **LandContent**. But the class **HighPlant** is a subclass of **LandContent**. Therefore the object property allow an individual from class **HighPlant** to relate with itself. Because we do not want this, we set the property to be **irreflexive**. The same situation we meet for the object property **hasBelow** and their inverse properties **isBehind**, **isBelow**.

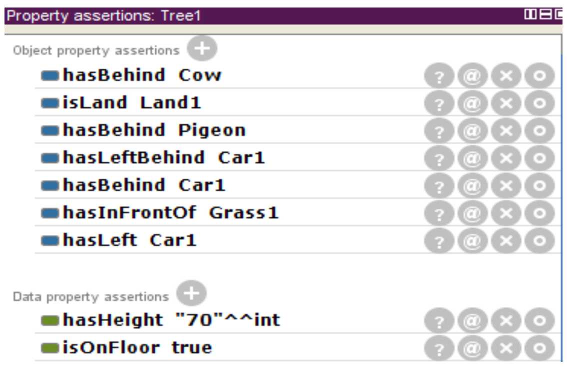
If (R(x,y)∧R(y, z))→R(x, z) the relationship is *transitive*.

$$\forall x,y,z\in\text{Content and }(\text{hasLeft}(x,y)\wedge\text{hasLeft}(y,z)) \rightarrow \text{hasLeft}(x, z) \quad (29)$$

If an object O1 **hasLeft** an object O2 and the object O2 **hasLeft** an object O3, then object O1 **hasLeft** object O3. We conclude that the property **hasLeft** is transitive.

C. Individuals

The ontology is not complete until the addition of individuals. For every class we can establish as many individuals as we want. For this ontology we considered that an individual or two for a class are enough. As the individuals are added we associate the appropriate object and data properties. In Fig. 3 there is an individual which is related through spatial relationships to another individuals.



Property assertions: Tree1

Object property assertions

hasBehind Cow
isLand Land1
hasBehind Pigeon
hasLeftBehind Car1
hasBehind Car1
hasInFrontOf Grass1
hasLeft Car1

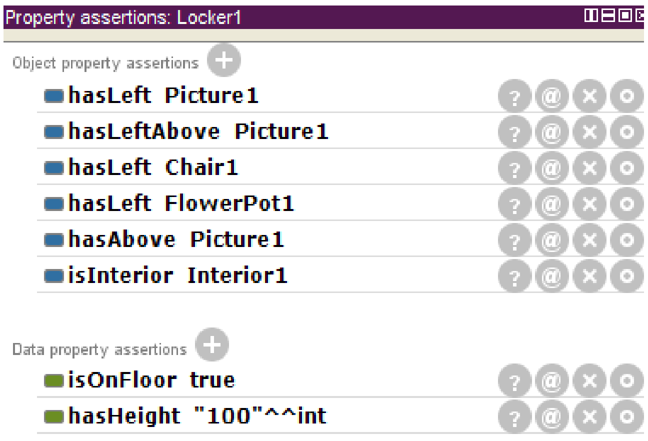
Data property assertions

hasHeight "70"^^int
isOnFloor true

Fig. 3. Properties of the individual named Tree1

Fig. 4, Fig. 5, Fig. 6 demonstrate that the relationship **hasLeft** is transitive.

Fig. 4 illustrates the properties of the individual **Locker1**. On its left side is positioned the individual **FlowerPot1**.


Property assertions: Locker1

Object property assertions

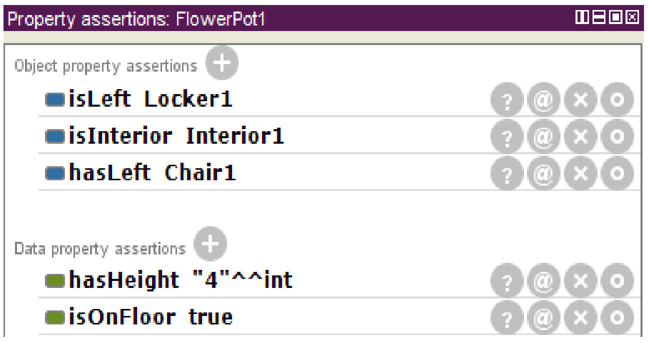
- hasLeft Picture1
- hasLeftAbove Picture1
- hasLeft Chair1
- hasLeft FlowerPot1
- hasAbove Picture1
- isInterior Interior1

Data property assertions

- isOnFloor true
- hasHeight "100"^^int

Fig. 4. Properties of the individual named Locker1

Fig. 5 illustrates the properties of the individual **FlowerPot1**. On its left side is positioned the individual **Chair1**.


Property assertions: FlowerPot1

Object property assertions

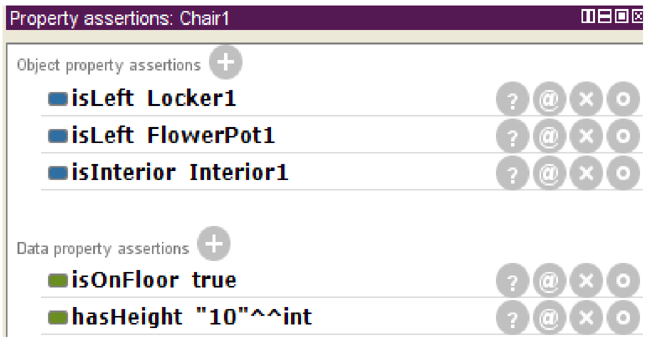
- isLeft Locker1
- isInterior Interior1
- hasLeft Chair1

Data property assertions

- hasHeight "4"^^int
- isOnFloor true

Fig. 5. Properties of the individual named FlowerPot1

Fig. 6 illustrates the properties of the individual **Chair1**. It is positioned on the left side of the individual **Locker1**, which demonstrates that the relation **hasLeft** is transitive.


Property assertions: Chair1

Object property assertions

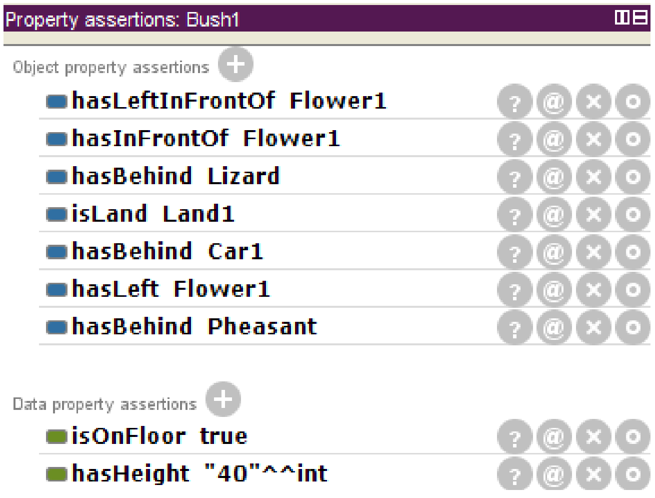
- isLeft Locker1
- isLeft FlowerPot1
- isInterior Interior1

Data property assertions

- isOnFloor true
- hasHeight "10"^^int

Fig. 6. Properties of the individual named Chair1

In Fig. 7 we can see that composing relationships **hasLeft** and **hasInFrontOf** we obtain a more precise positioning of the flower from the bush.


Property assertions: Bush1

Object property assertions

- hasLeftInFrontOf Flower1
- hasInFrontOf Flower1
- hasBehind Lizard
- isLand Land1
- hasBehind Car1
- hasLeft Flower1
- hasBehind Pheasant

Data property assertions

- isOnFloor true
- hasHeight "40"^^int

Fig. 7. Properties of the individual named Bush1

D. *Classifying the ontology*

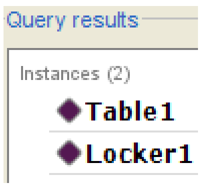
We use a reasoner to process the ontology. The reasoner verify the consistency of the ontology: it examines if there is no inferred in the class hierarchy and if every classes can have individuals [9]. We use HermiT, an open-source released under LGPL reasoner, to verify the ontology which is proved to be consistent.

With the reasoner started we apply DL Query on this ontology in order to prove that it is correctly created. In this article we present some examples of such queries.

In order to determine furniture which has above paintings we write:

$$\text{hasAbove some Picture} \tag{30}$$

Fig. 8 depicts the result of query (30).


Query results

Instances (2)

- Table1
- Locker1

Fig. 8. Furniture which has above paintings

In order to determine furniture which has above, but precisely at left side, we write:

$$\text{hasLeftAbove some Picture} \tag{31}$$

The result of query (31) is shown in Fig. 9.

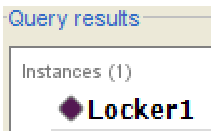


Fig. 9. Furniture which has left above paintings

In order to determine the land that has is his right side a particular water and has trees which has left behind a particular car, we write:

(hasRight value Water1) and (hasLand some (hasLeftBehind value Car1)) (32)

Fig. 10 illustrates the result of query (32).

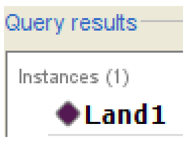


Fig. 10. The wanted land

IV. CONCLUSIONS AND FURTHER WORK

In this article we focus on the spatial relationships between ontology classes. We use the object properties to place objects on a scene. Being given a scene that contains an object, by choosing other objects will be generated some different scenes depending on the position of the new objects to the original object. This action is possible because of spatial relationships between elements.

As further work we intend to apply on this particular ontology a genetic algorithm in order to obtain automatically generated scenes. We obtain in this way a visual effect for a better understanding of evolutionary ontologies research.

Another further development is represented by the defining of temporal ontological relations for movies.

REFERENCES

[1] O. Âkerberg; H Svensson, B Schulz; P Nugues, “CarSim: an automatic 3D text-to-scene conversion system applied to road accident reports”, Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 2. Association for Computational Linguistics, 2003

[2] L Barazzetti, F Remondino; M Scaioni, “Automation in 3D reconstruction: results on different kinds of close-range blocks”, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXVIII, Part 5, UK. 2010

[3] S Bergen, B. J. Ross, “Automatic and interactive evolution of vector graphics images with genetic algorithms”, the Visual Computer, 2012, 28.1: 35-45

[4] S Colton, “The painting fool: Stories from building an automated painter”, Computers and creativity. Springer Berlin Heidelberg, 2012, 3-38

[5] C Costea, “Applications of multi-agent system technologies to power system”, International Symposium for Design and Technology of Electronic Packages, Baia Mare, 2007

[6] B Coyne, O Rambow, J Hirschberg, R Sproat, “Frame semantics in text-to-scene generation”, Knowledge-Based and Intelligent Information and Engineering Systems. Springer Berlin Heidelberg, 2010, 375-384

[7] N Drummond, M Horridge, H Knublauch, “Protégé-OWL tutorial”, In 8th International Protégé Conference, 2005

[8] T Gruber, “A Translation Approach to Portable Ontology Specication”, 1993

[9] M Horridge, H Knublauch, A Rector, R Stevens, C Wroe, “A Practical Guide To Building OWL Ontologies Using The Protg-OWL Plugin and CO-ODE Tools Edition 1.0.”, University of Manchester, 2004

[10] M Lobonțiu, A Petrovan, “Product development ontology(1). Information integration concepts”, Revista de Management și Inginerie Economică, ISSN 1583-624X, 4(46) vol. 11, 2012, 43-56

[11] S. A Malik, R. S Kunwar, M. E Haque, “Automatic image registration using evolutionary algorithm”, Recent Research in Science and Technology, 2012, 4.1

[12] O Matei, “Defining an Ontology for the Radiograph Images Segmentation”, 9th International Conference on Development and Application Systems, 2008

[13] O Matei, “Evolutionary Computation: Principles and Practices”, Risoprint, 2008.

[14] O Matei, “Ontology-Based Knowledge Organization for the Radiograph Images Segmentation”, Advances in Electrical and Computer Engineering, 8(15), 2008

[15] S McNicholas, E Potterton, K.S Wilson, M.E.M Noble, “Presenting your structures: the CCP4mg molecular-graphics software”, Acta Crystallographica Section D, Biological Crystallography, ISSN 0907-4449, 2011

[16] A Petrovan, M Lobonțiu, “Product development ontology. A case study, Quality Access to success”, ISSN1582-2559, S5 vol. 13, 2012, 393-398

[17] A Petrovan, G Lobonțiu, S Ravai-Nagy, “Broadening the Use of Product Development Ontology for One-o Products”, Applied Mechanics and Materials Vol. 371, 2013, 878-882

[18] L Tang, C Chen, J Zou, Y Lin, D Lin, J Li, “OntoPlant: an integrated virtual plant software package for different scale applications”, Spatial Data Mining and Geographical Knowledge Services (ICSDM), 2011 IEEE International Conference on. IEEE, 2011

[19] L. F Yu, S. K Yeung, C. K Tang; D Terzopoulos, T. F Chan, S Osher, “Make it home: automatic optimization of furniture arrangement”, ACM Trans. Graph., 2011