Non-commercial object-base scene description

Stephen Gulliver, Gheorghita Ghinea, Kulveer Kaur

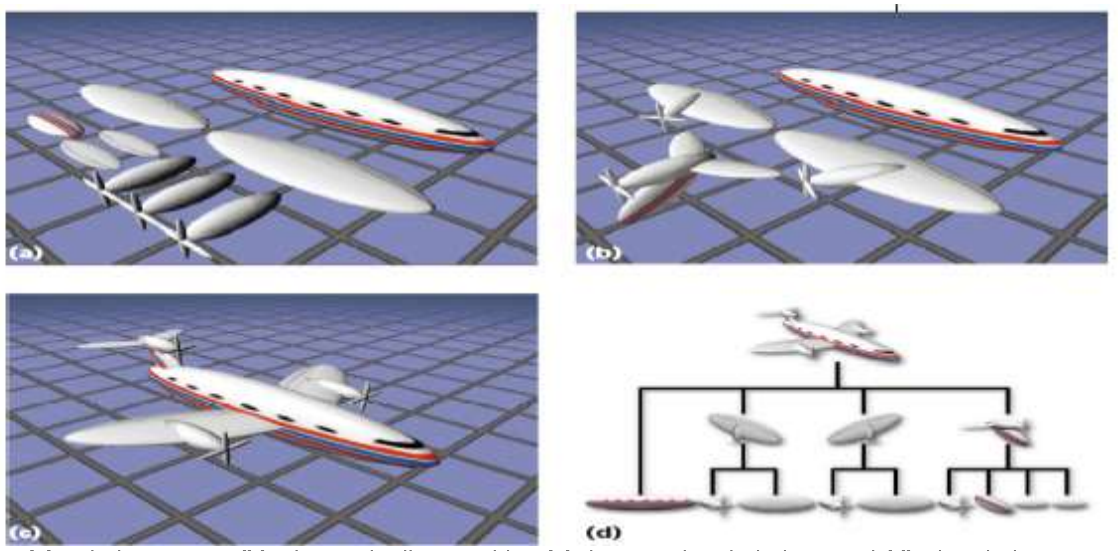
Brunel University, UK

Introduction

This paper investigates the development of a non-commercial object-based scene description language, for the effective capture of two- and three-dimensional scenes. Design science research techniques were applied to determine the factors involved in effective scene capture. The solution facilitates the description of 2D and 3D environments via interpretation of object relationships; the implementation of the inheritance, functionality, interactions and behaviour.

There have been a number of languages developed to create 2D and 3D scenes and virtual environments. Sadly the languages possess problems of interoperability, reliability and performance; the inability to incorporate effectively dynamic or interactive scenes (Walczak, 2003); as well as limited or conflicting commercial support of rendering within web browsers (Macdonald, *et al., 2005*). Modifications to previous standards have been made, yet many of these problems still exist.

The main technique used for describing 3D environments is via ***hierarchical scene graphs*** (Nadeau, 1999). Scene graphs are a hierarchy of groups and shapes arranged in a tree structure. Each of the parents and children within a scene graph are called nodes. These nodes are used to construct 3D shapes. Figure 1 shows the making of an aeroplane using a scene graph where all the parts are assembled one by one in order to make the final design (Nadeau, 1999).



(a) Airplane parts**, (b)** wing and tall assembles **(c)** the completed airplane and **(d)** The airplane scene graph Figure 1 Building a toy airplane using a scene graph [Nadeau, R. D. (1999), 19]

Our work emphasises upon describing both scene hierarchy, yet also how scene objects link together – interactivity both physical and functional. An office, for example, is defined by the effective placement of objects, such as a keyboard, a CPU, a mouse, a table / desk and chair. The table may be made up of other objects: drawers, legs, etc. This table is physically linked in some way to other objects in the environment, such as the floor, the keyboard, the monitor, etc. In addition, certain objects have definable functionality allowing interaction (both with the user, but also with other objects). Each object has a physical appearance, location and orientation, it may have considerable functional potential (often similar in nature to other instances of its type).

The design science approach was used in our work (see figure 2 for Design Science Approach flow diagram). Simple real world examples were used to establish the important factors involved in the effective capture of 2D and 3D scenes. Four iterations were performed to check all concepts, however all iterations were checked against test examples in order to ensure effective object capture. Abduction processing defined the following requirements.

1. The final solution should be able to define both 2- and 3-Dimensional scenes using a XML- based description.
2. An object should be able to inherit the properties of other objects.
3. The design should be able to add functionality to an object.
4. The design is needed to define the behaviour on the basis of properties.
5. The concept of interaction should be clearly defined.

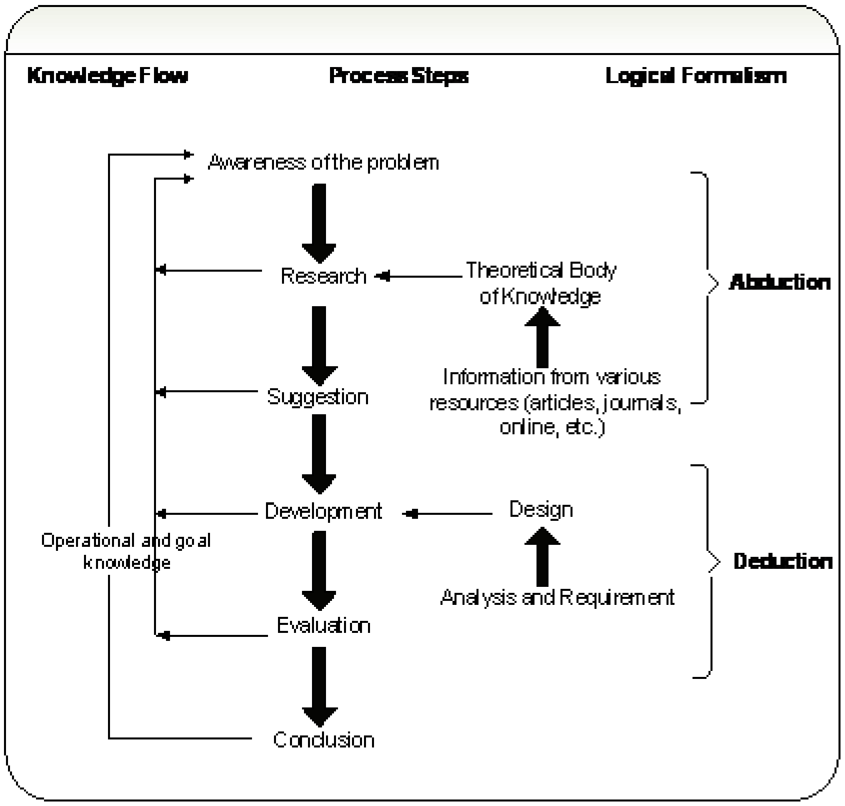
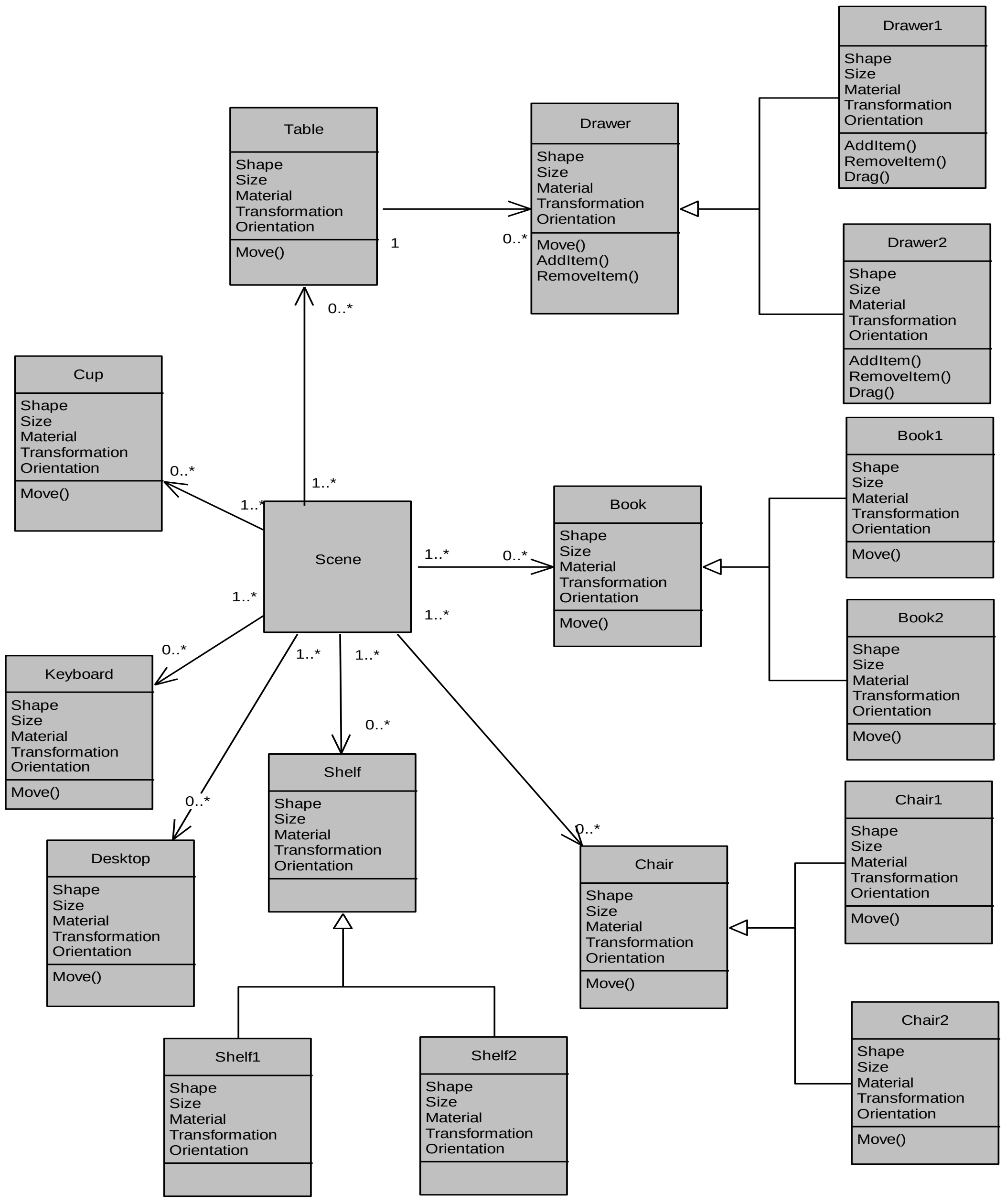


Figure 2 Design Science Approach flow diagram

Describing scenes and Objects

A scene is fundamentally the relationship that exists amongst objects. Every relationship has a meaning and it can be used to aid interactivity. Although the list of link definitions is unlimited, the set of relationships used to describe a scene in our example is limited to: *On the top of* (ObjA, ObjB); *Opposite to* (ObjA, ObjB); *In the RHS* (ObjA, ObjB); *In the LHS* (ObjA, ObjB); *Next to the* (ObjA, ObjB); *Touches* (ObjA, ObjB). The distance between objects can be defined at instantiation, for example: *On the top of* (ObjA, ObjB, Distance).

When implementing the office example, our solution considers both hierarchy of objects (see figure 3), but also the interaction capabilities and relative placement of objects (see figure 4). Inclusion of functional inheritance was added, via the launching of programming interfaces, to facilitate single and multiple inheritance.



Drawer1
Shape Size Material Transformation Orientation
AddItem() RemoveItem() Drag()
Table
Shape Size Material Transformation Orientation
Move()
Drawer
Shape Size Material Transformation Orientation
Move() AddItem() RemoveItem()
1
0..*
Drawer2
Shape Size Material Transformation Orientation
AddItem() RemoveItem() Drag()
0..*
Cup
Shape Size Material Transformation Orientation
Move()
Book1
Shape Size Material Transformation Orientation
Move()
0..*
1..*
1..*
Book
Shape Size Material Transformation Orientation
Move()
Scene
1..*
0..*
1..*
1..*
Book2
Shape Size Material Transformation Orientation
Move()
0..*
1..*
1..*
Keyboard
Shape Size Material Transformation Orientation
Move()
0..*
Shelf
Shape Size Material Transformation Orientation
0..*
Chair1
Shape Size Material Transformation Orientation
Move()
0..*
Desktop
Shape Size Material Transformation Orientation
Move()
Chair
Shape Size Material Transformation Orientation
Move()
Shelf1
Shape Size Material Transformation Orientation
Shelf2
Shape Size Material Transformation Orientation
Chair2
Shape Size Material Transformation Orientation
Move()

Figure 3 Office objects defined within the scene heirarchy

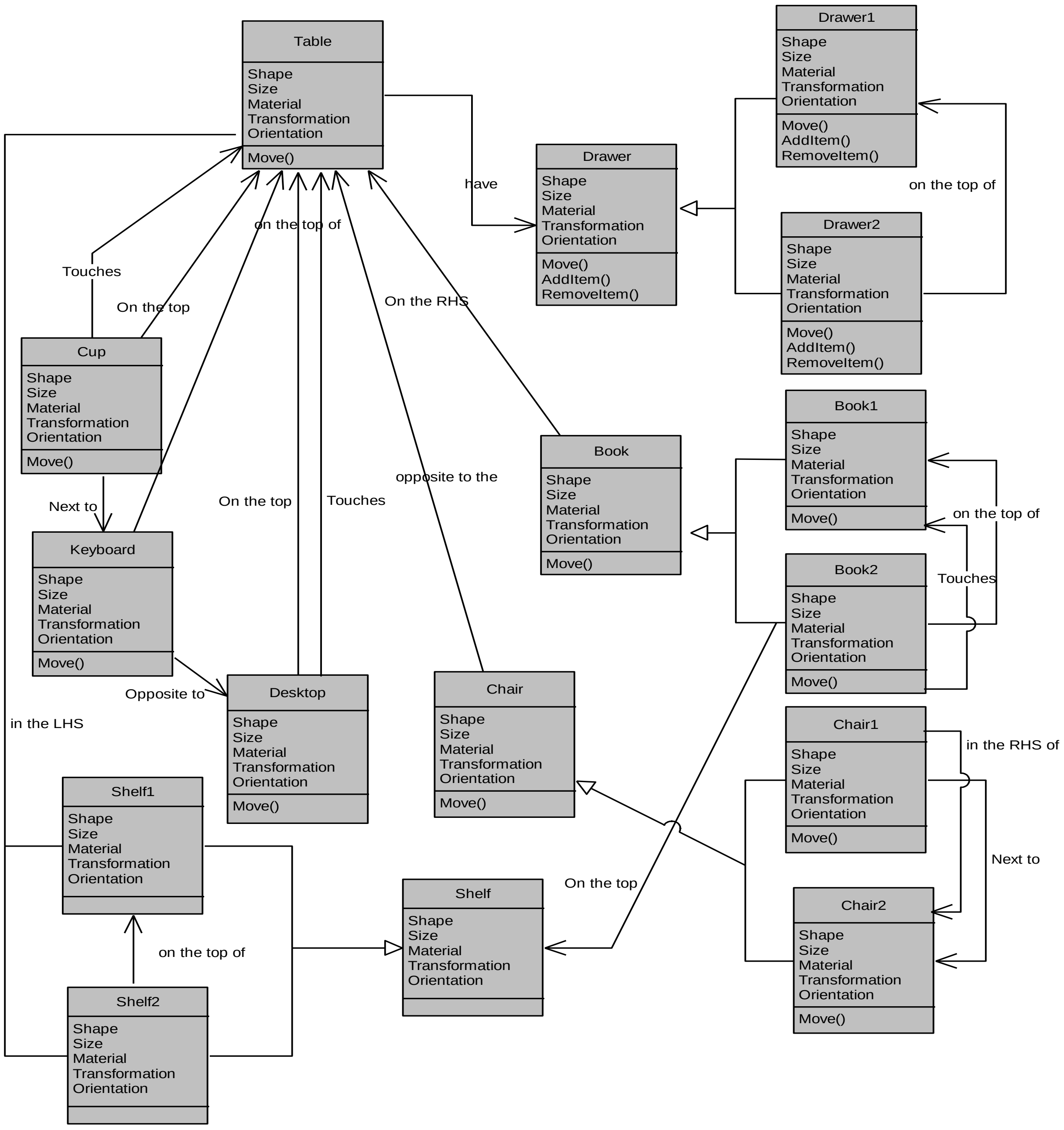
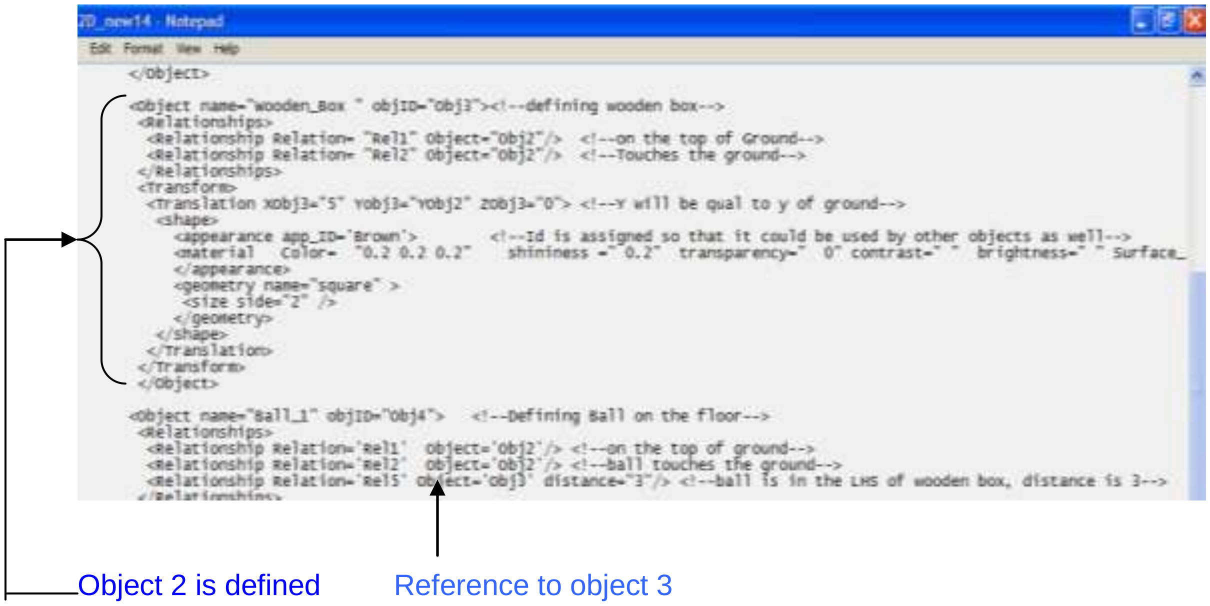


Figure 4 Office objects and their interactive relationships

Each relationship was given an identity so that it could be used by each of the objects (see figure 5). All information about relationships and locations of objects was described. A file link was used to describe the image appearance.

Shape / appearance of objects was described using current standardised modelling languages; size / scale, orientation and object material / texture was manipulated in object parameters. Documentation of functionality and inheritance was achieved by launching a java-based programming environments and using inherent java inheritance. An XML extension is being considered.



```
    </Object>

    <Object name="wooden_Box " objID="Obj3"><!--defining wooden box-->
     <Relationships>
      <Relationship Relation= "Rel1" Object="Obj2"/>  <!--on the top of Ground-->
      <Relationship Relation= "Rel2" Object="Obj2"/>  <!--Touches the ground-->
     </Relationships>
     <Transform>
      <Translation xObj3="5" yObj3="YObj2" zObj3="0"> <!--Y will be qual to y of ground-->
       <shape>
         <appearance app_ID='Brown'>       <!--Id is assigned so that it could be used by other objects as well-->
         <material   Color=  "0.2 0.2 0.2"    shininess =" 0.2"  transparency="  0" contrast=" "  brightness=" " Surface_
         </appearance>
         <geometry name="square" >
          <size side="2" />
         </geometry>
       </shape>
      </Translation>
     </Transform>
     </Object>

    <Object name="Ball_1" objID="Obj4">   <!--Defining Ball on the floor-->
     <Relationships>
      <Relationship Relation='Rel1'  Object='Obj2'/> <!--on the top of ground-->
      <Relationship Relation='Rel2'  Object='Obj2'/> <!--ball touches the ground-->
      <Relationship Relation='Rel5' Object='Obj3' distance="3"/> <!--ball is in the LHS of wooden box, distance is 3-->
     </Relationships>
```

Figure 5 Reference made to access relationships and objects through their Ids

Conclusion

The work describes the steps taken to implement a solution for scene description on the basis of relationships that exist amongst objects. The solution, via inclusion of other technologies, implements inheritance, addition of functionality. Moveover, manipulated properties are capable of defining interaction and behaviour of objects in a very well manner.

References

1. Macdonald, J.A., Brailsford,F.D.,Bagley,R.S.(2005) Encapsulating and manipulating Component object graphics (COGs) using SVG, ACM
2. Nadeau, R. D. (1999) Building Virtual Worlds with VRML, IEEE Computer Graphics and Applications
3. Walczak, K., Cellary, W. (2002) Building database Applications of Virtual Reality with X-VRML, ACM, Web 3D'02, Tempe, Arizona, USA, February 24-28