Creating Artistic Compositions Using Coalition-Forming Intelligent Agents

Alex Pytel[†]

David R. Cheriton School of Computer Science
University of Waterloo

Technical Report CS-2008-23

October 15, 2008

[†]email: apytel@cs.uwaterloo.ca

Abstract

Simulation of multi-agent systems (MAS) can be an effective technique for generating randomized compositions out of primitive picture elements. This report presents a complete implementation of a MAS for creating rule-based hierarchical arrangements of 3D objects in a plane and identifies some associated technical challenges, such as attaining good performance and usability.

1 Introduction

When viewed as the problem of creating compositions of pre-defined elements, procedural generation of virtual environments can be accomplished as a product of simulating a *multi-agent system* (MAS). In this context, the *agents* are primitive constructs consisting of local state and an objective function which determines their actions. The behaviour of the agents is framed primarily in terms of their ability to join and leave agent groups, called *coalitions*. This characterization of MAS is due to Mason et al. [2], who have shown that the placement of graphical elements in a composition based on the behaviour of the agents of a MAS can benefit from the emergent behaviour of the agents and make the produced composition appear to be a *gestalt* — a whole that is greater than the sum of its parts.

This report presents a multi-agent system designed to create randomized compositions of 3D primitives in a plane. These compositions can have a recursive organization and can arrange their elements in 2.5D in order to better utilize the 3D space. The emergent hierarchy of the agent coalitions forms the structure of the resulting scene as a hierarchical tree of transformations applied to the primitives. The framework allows for easy experimentation with agent behaviour by providing a common structure for adding new types of agents.

2 Related Work

Mason et al. [2] use the kind of MAS described above to generate 2D artistic compositions whose elements are created and arranged as the agents form coalitions. The process relies on the agents' emergent behaviour as they follow a set of rules that cause them to explore the agent space and interact with each other. The primary goal of the agents is coalition formation, which brings them closer to becoming an element of the displayed picture. The structure that the agents create emerges in a bottom-up manner, since the agents only have a subjective view of their environment. Demonstrations of the use of Surreal, which is an implementation of such an agent system, suggest that it is flexible in producing virtually any kind of composition. In particular, it was used to produce a Piet Mondriaan-like composition and a *seigaiha* (blue ocean waves) pattern [2].

There are other applications that benefit greatly from this ability of the agents to generate emergent structure based only on a subjective understanding of their environment. For example, stroke-based rendering, an NPR technique that generates an image by combining elements larger than a pixel (usually by following the pattern of a source image), has been successfully accomplished using a MAS [3]. In particular, the emergent behaviour of the agents can be used to effectively locate edges in an image, as well as place patterns of stipples or hatches. In a similar way, a MAS-based approach can be used to effectively determine a set of routes for navigating a given environment, as it has been shown by an algorithm inspired by the behaviour of biological bees [1].

A key difference between the three MAS applications is the degree of abstraction of the MAS environment from the environment of the underlying problem that the MAS simulation is intended to solve. In the 2D composition example [2], the agent environment is abstract (a hypertorus) and the coalition formation paradigm is used to relate it to the environment of the problem. In the stroke-based rendering example [3], the agent environment is almost the same as the concrete environment of the problem (which is the image to be rendered). In particular, the agent space is the input image augmented with additional information, such as object masks and temporary buffers for indirect inter-agent communication. Finally, in the bee-based navigation example, both of the environments are the same model of a physical world [1] requiring exploration.

3 Technical Details

3.1 Agent Interpretation

The MAS system presented in this report adopts the model of a separate completely abstract agent environment, as the most general one. Additionally, it also relies on framing the emergent agent-space structure in terms of the coalition formation paradigm used by Mason et al. [2], since that framework is most immediately suitable for translation into scene-space structure. These ideas establish the working principles of the MAS example, except for some details of agent usage and behaviour discussed below.

Although the agents and their space are entities which are abstracted, or not directly expressed in the scene-space environment, they are still given meaning through the agent coalition paradigm, which relates structure in the agent space to structure in the scene space. In particular, the objects expressed in the final scene are organized hierarchically, using a tree of transformations. For example, a group of objects that are equally spaced around a circle can contain both other such groups and primitives, which are the leaf nodes in the hierarchy. For a given object hierarchy to emerge in the scene, it is necessary for the agent coalitions to follow an identical pattern of containership in agent-space. This is illustrated in Figure 1 for the case of a CIRCLE, which is a circular arrangement of primitives, and a STACK, which is a vertically stacked arrangement of primitives; a primitive can be a cube, a pyramid, or a sphere.

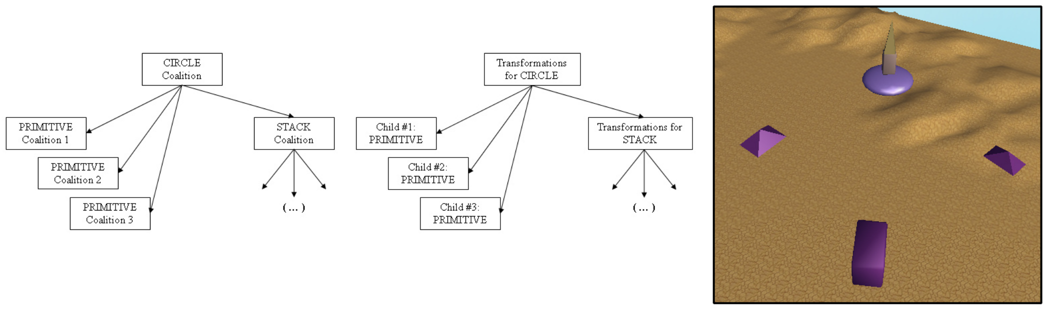

Figure 1: The structure of agent coalitions on the left is in 1-1 correspondence with the structure of the object hierarchy in the middle. The right panel shows the produced scene.

In addition to encoding structure, the agents can also represent certain specific types of information related to the scene objects, or their groups. For instance, there are agents for the type, size, and colour of the primitives. Additionally, there are intermediate coalitions of agents which contain partial information about a potential scene-space structure. These additional types of agents serve a technical role and are listed in Appendix A.

3.2 Agent Behaviour

To facilitate the agent to scene translation of structure, coalitions are themselves represented by instances of appropriate agent types which have the ability to become coalition *directors*. In this way, the behaviour of the entire simulation is essentially decided by the rules of joining or leaving an agent sub-tree administered by a coalition director. Each agent, the directors included, has a subjective understanding of its environment which consists of nearby agents in the surrounding agent space and some agents of its agent tree.

At each simulation step each agent evaluates its objective function in its current state as well as in the states corresponding to the agent's leaving its current coalition (if any) for each candidate coalition in its subjective environment; the result determines the actions taken by the agent (Appendix B contains a detailed list of actions). There are some fundamental rules that govern the process:

Rule I: Information Flow. The coalition director should have access to all the information available to its members [2]. In particular, this means that a director agent can search the neighbours of any node in its agent tree for another coalition to join. The other way around, when one agent attempts to join a second agent which belongs to a coalition, the first agent can look for other potential join candidates in that coalition.

Rule II: Decisions. All decisions made by the agents to undertake a particular action (joining or leaving a coalition) are based on the change to the objective function of the relevant agents. For example, leaving agent A for B can only be done if it increases B's objective more than it decreases A's.

Rule III: Altruism. Agents assume that whatever is good for the coalition they are joining is also good for them. In other words, agents seek the coalitions whose objective function they can increase the most by joining (the reciprocal objective increase is not defined).

Furthermore, an agent's objective function itself implicitly imposes some constraints specific to the type of the agent. Specifically, it is computed as a weighted sum of the results of a series of goal tests, such as the ones shown below. More information is provided in Appendix C.

Goal Tests		
Name	*Returns*	*Test*
member_parts	number	Member number (more is better)
min_complete	boolean	Minimum number of members present
specific_parts	number	Number of members which are mandatory
scene_express	boolean	Ready for scene expression

4 Results

Figure 2 shows one example of the dynamics of coalition formation in agent-space. In frame 1 a PRIMITIVE agent (P) is collecting information necessary for expressing a single primitive in the scene: COLOUR (c), one of possible type-communicating agents (t), and SCALE (s). In frame 2, the coalition formed by the PRIMITIVE agent is considered ready for expression in scene-space (to reflect this, it is marked with a double frame). The PRIMITIVE agent acts as a director for its coalition and searches its agent-space locality for another coalition it could join. Upon finding a STACK coalition (possibly by finding one of its children first and then going to the parent), which represents a number of primitives stacked vertically on top of each other, the PRIMITIVE coalition joins the STACK in frame 3. Additionally, the STACK gains a POSITION (x) member, and the STACK's original PRIMITIVE member gains a COLOUR.

In frame 4, the STACK needs another expressible PRIMITIVE in its coalition in addition to the members it has already (the two expressible PRIMITIVE agents and the POSITION agent) in order to be expressible itself. Therefore, when a newcomer PRIMITIVE agent considers whether to join the STACK coalition or a nearby incomplete ROW coalition (representing elements arranged in a row), it decides in favour of the former, observing that

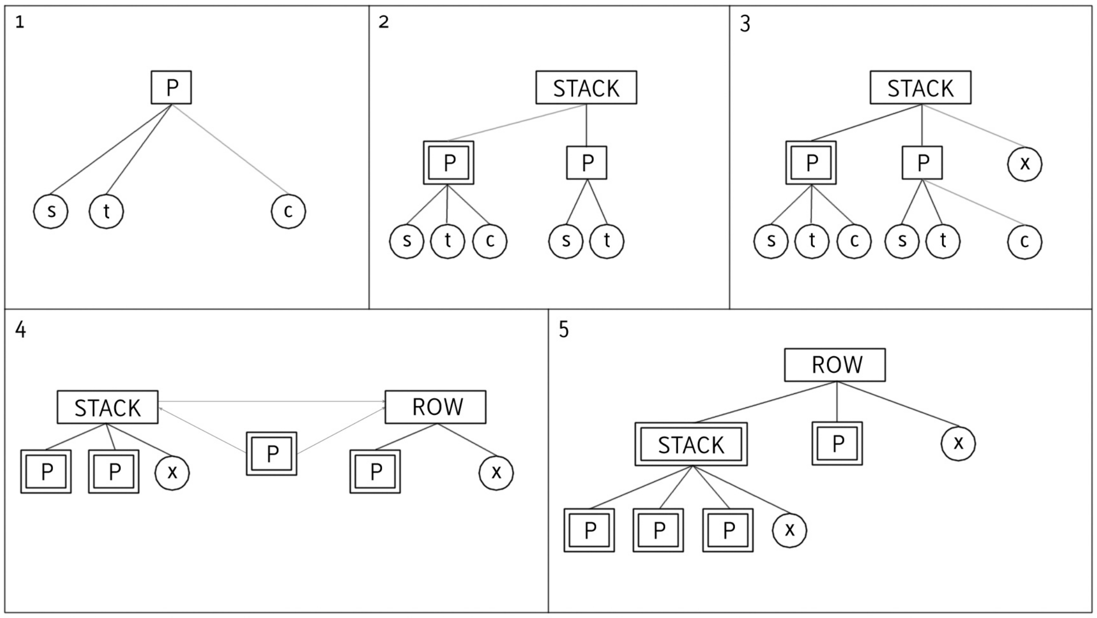

Figure 2: A possible way in which agent-space structures can evolve. Double-framed elements are considered ready to be expressed in the scene. Gray lines show possible linkages being evaluated.

joining the STACK will benefit the STACK more than joining the ROW will benefit the ROW. On the other hand, joining a coalition like the ROW one will strictly increase the STACK's objective function, so it joins the ROW right away. Frame 5 illustrates the result; eventually it could resemble Figure 4. Note that frames 4 and 5 omit the members of PRIMITIVE coalitions for clarity. Also note that the agents are capable of other operations, such as leaving coalitions (the full list of possible operations is presented in Appendix B) and that the example omits some optional agent types (listed in Appendix A).

The MAS approach to creating compositions of elements in a 3D environment has worked well for simple scenes. Figure 3 shows the simplest non-recursively organized example of an agent coalition (called UNIT) being expressed in the scene. For a better example, Figure 4 shows a recursively organized structure of the kind discussed in Section 3.1 — this one involving agent coalitions of types ROW, STACK, and some PRIMITIVE types.

Due to the MAS framework the system scales extremely easily (Figure 5). However, the nature of the simulation is such that a lot of operations need to be tried and failed by each agent at each iteration. This is a fundamental source of inefficiency when there are a lot of candidates for each operation. Therefore there can be an unacceptable degradation of performance when increasing the density of agents in the agent space or running the simulation for too many iterations, which results in unmanageably large coalition trees.

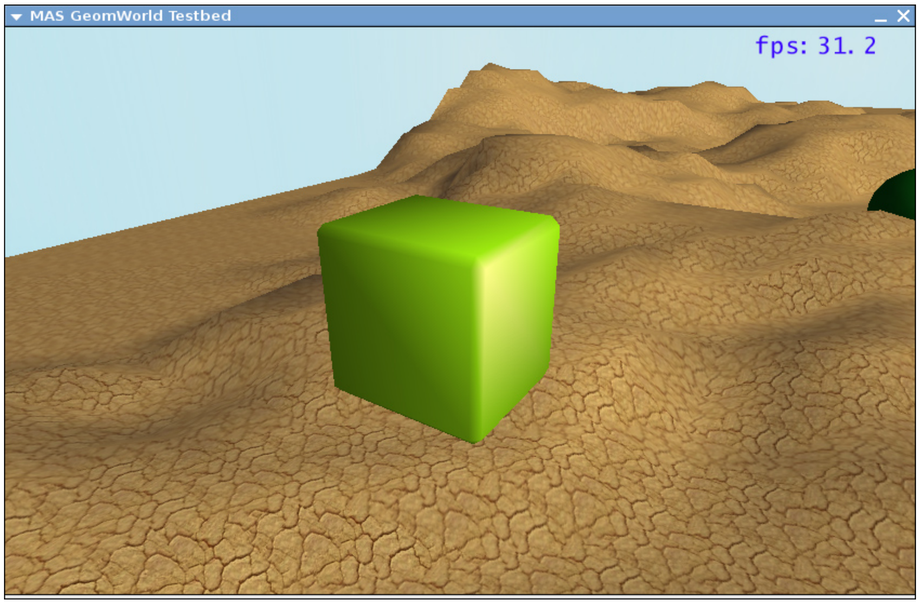

Figure 3: Expression of a UNIT — the simplest composition element.

It is also of concern that the extra computation time required in such pathological situations, benefits the resulting scene composition very little. One indication of the problem is that there is typically a large number of agents that take a long time to collect enough information to be expressed in the scene. These agents are a dead weight that is temporarily useless for the production of the final composition, but contributes heavily to the size of coalition trees. For example, the test run whose result is shown in Figure 5 has an average tree size of 60 agents for CIRCLE agents (responsible for creating a circular arrangement of primitives), but the largest number of primitives placed in a circle is only 4 in the picture. The discrepancy is attributable to there being only 293 expressible PRIMITIVE agents (responsible for defining the properties of a primitive) out of a total of 452.

Therefore, it is unsurprising that experimentation with pruning candidate lists and limiting descent (or ascent) in especially large agent subtrees has shown that the ability of the MAS to create complex organization (in the agent tree or, equivalently, in its visualization) benefits very little from the complexity of the rules and behaviours encoded into the agents. To conclude, the system sometimes spends time on dealing with details that do not help in making quick progress to the solution of the problem.

An additional concern related to the useability of the system is that the top-down hierarchical nature of the transformations in a typical 3D scene goes counter to the bottom-up organization of the agents. This has been a challenge when developing and working with the system.

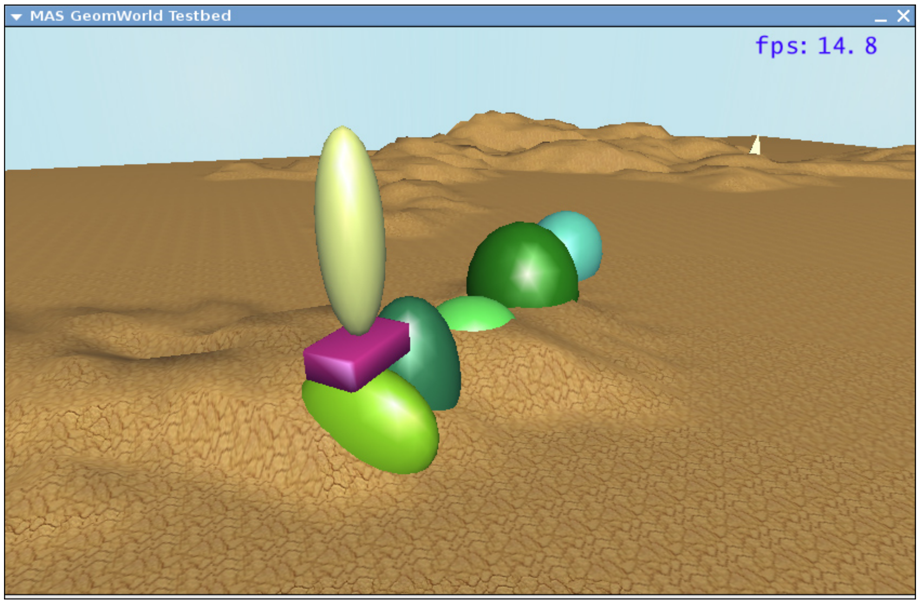

Figure 4: A ROW with a STACK inside it.

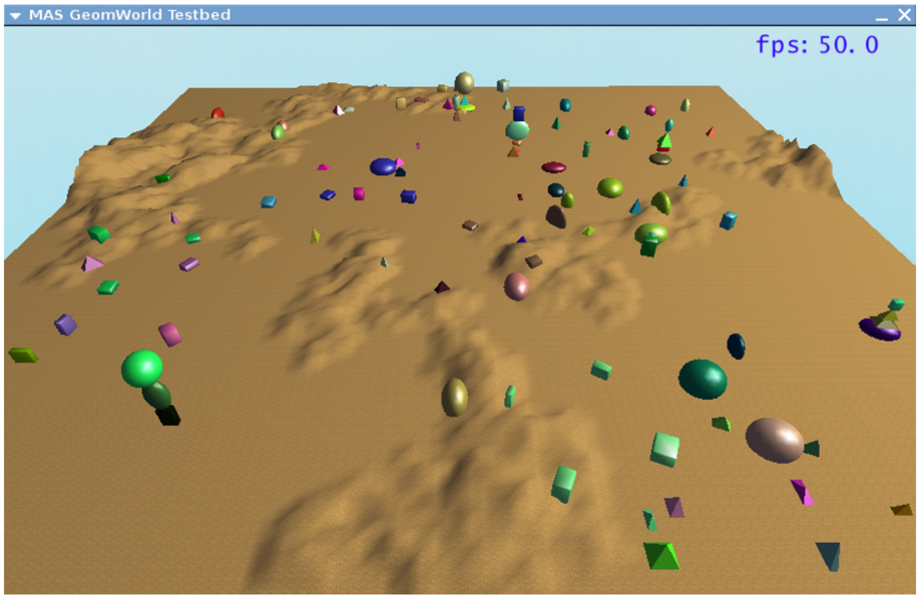

Figure 5: Testing the system with 2000 initial agents and 778 agents that were spawned during the simulation.

5 Conclusion

The selected approach to implementing a MAS framework for creating element compositions has both strengths, particularly conceptual ones, and weaknesses. Specifically, the abstraction of agents and agent space from the problem environment and the associated coalition paradigm have been helpful for reasoning about the design of the system. However, the implementation has exhibited a number of inefficiences, indicating that not all of the design decisions have perfectly fit the problem.

Still, experimentation with the system has shown that the basic framework has a lot of merit as an approach for procedural generation of environments out of pre-defined elements. First, it is suitable for creating complex rule-based organization of elements with an emergent structure. Second, a major advantage of MAS over such approaches as a straightforward recursive construction of the scene is that the rules guiding the simulation and the behaviour of the agents can be used to produce many variations of the basic expression theme with minimal changes.

APPENDIX

A Agent Types

The following are the agent types that have been used with the system to produce the results described in the report:

Agent: **SPHERE**	Contains: type information
Required to express a sphere primitive in the scene.	

Agent: **BLOCK**	Contains: type information
Required to express a block primitive in the scene.	

Agent: **PYRAMID**	Contains: type information
Required to express a pyramid primitive in the scene.	

Agent: **POSITION**	Contains: randomly generated position information
Determines a coalition director's position, which serves as a reference point for transformations acting on the primitives expressed by its agent tree.	
Note: Represents absolute position, not offset; therefore only used by the topmost director, or by an agent whose director is not expressible.	

Agent: **ROTATION**	Contains: randomly generated rotation information
Determines an agent's rotation, which can act on the primitives expressed by the agents in its (agent) subtree, if any.	
Note: This is an optional local rotation around the Y-axis; it is always used when available.	

Agent: **SCALE**	Contains: randomly generated x, y, and z size
Determines the scale of a primitive if an agent expresses one.	
Note: Only applied at the leaves of the scene tree to scale the primitives.	

Agent: **COLOUR**	Contains: randomly generated colour
Determines the colour of a primitive if an agent expresses one.	
Note: Colour information does not propagate through the scene tree.	

Agent: **PRIMITIVE**	Contains: • One of {SPHERE, BLOCK, PYRAMID} • SCALE • COLOUR • ROTATION
Brings together the information needed to express a primitive in the scene.	
Note: Acts as an intermediate construction which is not expressible by itself.	

Agent: **UNIT**	Contains: • PRIMITIVE • POSITION
Expresses a solitary, optionally rotated, scaled, coloured primitive in the scene.	
Note: The most basic composition element; not hierarchical (can not be incorporated into a more complex recursive pattern).	

Agent: **STACK**	Contains: • list of PRIMITIVEs • POSITION
Expresses a stack of primitives in the scene.	
Note: This is a *single-level construction*: it can be incorporated into other patterns, but can not contain anything other than primitives itself.	

Agent: **ROW**	Contains: • list of members: {PRIMITIVE, ROW, CIRCLE, STACK} • POSITION • ROTATION
Expresses a row of objects in the scene, where each object can be a primitive or another composition element (such as a CIRCLE, or a STACK).	
Note: This is a *multi-level construction.* POSITION determines the absolute position of the first object in the row; ROTATION determines the rotation of the row "line" about that point.	

Agent: **CIRCLE**	Contains: • see ROW
Expresses a composition of objects that are evenly spaced around a circle.	
Note: This is a *multi-level construction* similar to a ROW.	

B Agent Actions

The following are the fundamental actions that the agents can perform during the simulation:

Operation: **registration**	Requirements: none
Registers an agent with a cell in the agent space to facilitate neighbourhood queries.	
Note: All agents must do this.	

Operation: **random walk**	Requirements: • the agent has no director • there is nothing else to do
The agent random walks the agent space, looking to encounter other agents.	
Note: Directors also update the position of other agents in their tree.	

Operation: **exploration**	Requirements: depend on the operation that prompted the exploration.
The agent obtains a list of candidates to perform some other operation on them.	
Note: Implements Rule I.	

Operation: **join attempt**	Requirements: • the agent has no director • the agent is not a zombie
The agent finds the best candidate coalition to join.	
Note: Implements Rules I — III.	

Operation: **sanctioned join attempt** (leave and join)	Requirements: the agent is sanctioned by its (topmost) director to attempt to leave for a better group if one is available.
During this operation the agent only uses the information that it would have if it were on its own (Rule I). It also must consider the possibility that it leaves its parent childless upon departure, in which case the parent must be exorcised from the tree and become a zombie as an agent population ratio control mechanism.	
Note: Implements Rules I — III.	

Operation: **spawn attempt**	Requirements: • the agent has no director • the agent is not a zombie • a previously performed join attempt has failed
The agent performs an exploration and selects another director-less non-zombie agent. Then the types of the two agents determine the type of the new agent that is spawned (and whether it is created at all), and the two old agents join its tree immediately.	
Note: This is a special operation introduced to simplify population ratio control (more complex agent types are spawned if they are needed, if there are too many spawned, they die off).	

Operation: **move**	Requirements: the agent has no director.
The agent gives its children a chance to break free for another coalition and exorcises any resulting dead (including itself). If it is still alive following this operation, it carries out a *join attempt*. If that fails, it tries a *spawn attempt*. If that fails, too, the agent *random walks*.	
Note: ties all the other operations together.	

C Agent Goals

The following lists the goal tests making up the objective function of each agent type:

Goal Tests		
Name	*Returns*	*Test*
member_parts	number	Member number (more is better)
min_complete	boolean	Minimum number of members present
specific_parts	number	Number of members which are mandatory
scene_express	boolean	Ready for scene expression

Agents	*Goals*	*Notes*
SPHERE, BLOCK, PYRAMID, POSITION, ROTATION, SCALE, COLOUR	None	Agents representing basic properties are meant to communicate information.
PRIMITIVE	• specific_parts • scene_express	All members are equally valuable; presence of 4 members is required (but not sufficient) for expression.
UNIT	• specific_parts • scene_express	
STACK, ROW, CIRCLE	• member_parts • min_complete • specific_parts • scene_express	For a more complicated construction, it is important to have a minimum number of members. For the topmost director or an agent whose director is not expressible having many members, but no position element is not very valuable. Therefore, more weight is given to position being present using test specific_parts.

References

[1] N. Lemmens, S. de Jong, K. Tuyls, and A. Nowe. A Bee Algorithm for Multi-Agent Systems: Recruitment and Navigation Combined. In *Proceedings of ALAG, an AAMAS workshop*, 2007.

[2] K. Mason, J. Denzinger, and S. Carpendale. Negotiating Gestalt: Artistic Expression by Coalition Formation between Agents. In *5th International Symposium on Smart Graphics*, Frauenwoerth Cloister, Germany, August 2005.

[3] S. Schlechtweg, T. Germer, and T. Strothotte. RenderBots — Multi Agent Systems for Direct Image Generation. *Computer Graphics Forum*, 24:283–290, 2005.