



A shape grammar approach to computational creativity and procedural content generation in massively multiplayer online role playing games

Kathryn E. Merrick^{a,*}, Amitay Isaacs^a, Michael Barlow^a, Ning Gu^b

^a School of Engineering and Information Technology, University of New South Wales, Australian Defence Force Academy, Northcott Drive, Canberra, 2600 ACT, Australia

^b School of Architecture and Built Environment, University of Newcastle, Australia

ARTICLE INFO

Article history:

Available online 17 October 2012

Keywords:

Computational creativity
Interest
Novelty
Shape grammars
Massively multiplayer
Online role-playing games
Instances
Procedural content generation

ABSTRACT

With rapid growth in both production costs and player populations over the last decade, the computer games industry is facing new scalability challenges in game design and content generation. The application of computers to these tasks – called procedural content generation – has the potential to reduce the time, cost and labor required to produce games. A range of generative algorithms have so far been proposed for procedural content generation. However, automated game design requires not only the ability to generate content, but also the ability to judge and ensure the novelty, quality and cultural value of generated content. This includes factors such as the surprise-value of generated content as well as the usefulness of content in the context of a particular game design. Studies of human designers have identified that the ability to generate artefacts that are novel, surprising, useful and valuable are facets of the human cognitive capacity for creativity. This suggests that computational models of creativity may be an important consideration for developing tools that can aid in or automate design processes. However, such cognitive models have not yet been widely considered for use in procedural content generation for games. This paper presents a framework for procedural content generation systems that use computational models of creativity as a part of the generative process. We demonstrate an example of such a system for generating instances for massively multiplayer, online role-playing games. The system combines the generative shape grammar formalism with a computational model of interest based on the Wundt curve to select new designs that are similar-yet-different to existing human designs. The approach aims to capture the usefulness and value of an existing human design while introducing novel or surprising variations through the model of interest. The system incorporates a metric that permits generated designs to be evaluated in terms of both their similarity to human designs and their novelty in the context of existing designs.

© 2012 International Federation for Information Processing Published by Elsevier B.V. All rights reserved.

1. Introduction

Procedural content generation (PCG) is the programmatic generation or adjustment of the content of a computer game. Various aspects of PCG have been studied including online and offline PCG; PCG of different types of game content; and different classes of algorithms for PCG [1,2]. One of the key challenges for all kinds of PCG lies in the need not only to generate game content, but to judge and ensure the novelty, quality and cultural value of generated content. This includes factors such as the surprise-value of generated content as well as the usefulness of content in the context of a particular game design. This paper proposes an approach to this issue by presenting a framework for PCG that generates new designs in a manner that can capture the usefulness and value of existing designs while also introducing novel or surprising

variations. Specifically, we describe, demonstrate and evaluate an example of such a system for generating instances for massively multiplayer, online role-playing games (MMORPGs).

Studies of human designers have identified that the ability to generate artefacts that are novel, surprising, useful and valuable are facets of the human cognitive capacity for creativity [3]. Thus, to achieve these aspects in a PCG framework we draw on models of design cognition and creativity in humans and artificial systems. The approach in this paper provides a framework for building PCG tools that can take components of high quality designs produced by human designers and creatively recombine, reuse and augment those components using artificial creative processes to generate new designs. The approach aims to incorporate two broad aspects of creativity. First, usefulness and value are captured in a new design by ensuring that there is similarity or ‘closeness’ between new designs and one or more existing human designs. Secondly, novel and surprising variations are introduced in new

* Corresponding author. Tel.: +61 2 6268 8023; fax: +61 2 6268 8443.

E-mail address: k.merrick@adfa.edu.au (K.E. Merrick).

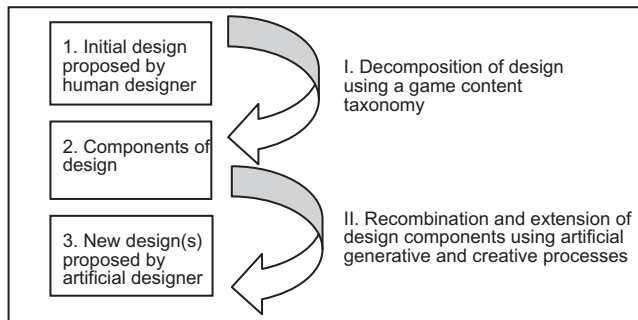


Fig. 1. A framework for computational creativity in procedural content generation.

designs using a computational model of interest to evaluate new designs.

Fig. 1 illustrates the high level framework for computational creativity in PCG proposed in this paper. First, to permit new designs to capture aspects of the usefulness and value of high quality human designs, our framework assumes the existence of at least one initial human design as a starting point for the creative process. This design is then decomposed (Step I) into primitive components that become the design variables input to the creative process (Step II). The creative process recombines those components to create new designs. This paper presents a tool that aids the human designer in Step I and automates the role of the human in Step II.

The remainder of this paper is organized as follows. Section 2 provides a cross-disciplinary review of literature in creative design and PCG. In Section 2.1 a number of generative algorithms and their role in PCG are briefly discussed. A detailed review of shape grammars is provided as background for the specific approach taken in this paper. As a basis for decomposing human designs into their constituent elements, Section 2.2 reviews existing taxonomies of the elements of game design. Section 2.3 reviews the elements of MMORPGs in particular, as the focus for the work in this paper.

Section 3 selects a subset of the elements of game content for MMORPG instances as the basis for a formal specification of our framework for computational creativity in PCG. Section 4 demonstrates an implementation of this framework and evaluates a set of designs for MMORPG instances it produces, including the design of the game space and the placement of non-player characters (NPCs) and interactive objects within the space. The paper concludes in Section 5 with a discussion of the implications and future directions of this work.

2. Background and literature review

The link between PCG in computer games and creative design lies in the recognition that PCG systems cannot simply generate new game content, but must also select only appropriate designs from generated content. Likewise, it has been recognized that creative design is concerned with more than just the introduction of something new into a design. While novelty is considered necessary for creativity, the introduction of ‘something new’ should also lead to a generated design that is somehow unexpected (surprising) as well as being useful, valuable or appropriate [4]. This link suggests that PCG systems must incorporate both generative and creative components. This assumption forms the foundation of the work in this paper. The following sub-sections thus review literature from both design cognition, and PCG to provide the background for the work in this paper.

2.1. Creativity and design

PCG can be thought of as automated design of all or part of a computer game. Broadly speaking, game design is the process of designing the environment, content, storyline and rules of a game. It can include the design of 3D objects and textures for buildings, terrain and NPCs as well as the design of dialog, scenarios, puzzles and stories. Design in a more general sense can be described as a process of purposeful, constrained decision-making, exploration and learning [4]. A design task – such as designing a character or building – can be conceived of as a problem space described by a set of variables. Decision-making processes select values for these variables from a solution space; exploration changes either the problem space or the solution space; and learning implies a restructuring of knowledge to reduce the distance between the problem and solution spaces.

The role of creativity in design has been studied from a number of perspectives [3]. From one perspective, creativity is considered to reside only in the generated design as it is evaluated by an individual or a society of users. From another perspective creativity is considered as a process that produces designs that may be evaluated as creative. This latter perspective informs research that seeks to produce artificial systems that can exhibit such creative processes, and is thus relevant to this paper.

A range of generative design approaches have so far been proposed for PCG in computer games. These approaches include pseudo-random number generators, generative grammars [5,6], image filtering, spatial algorithms, complex systems simulations and artificial intelligence algorithms [2]. There is currently some overlap of these approaches with those used in artificial systems that automatically generate creative design solutions based on models of creativity. However, the latter systems tend to focus on modeling the biological and cognitive phenomena associated with creativity, rather than on design generation alone. Examples of biologically inspired models include evolutionary systems and genetic algorithms [7,8], while examples of cognitively inspired models include analogy [9,10], novelty, interest and curiosity [11], situated reasoning [12], flexible ontology [13] and case-based reasoning [14,15]. These tools can be fully automated, or require interaction from [16] or collaboration among [17] designers.

The demonstration in this paper is an interactive tool that incorporates a model of creativity based on novelty and interest with a shape grammar system for PCG. These models are described in detail in the next two sections.

2.1.1. Computational models of novelty and interest as a basis for modeling creativity

Computational models of creativity – and associated models of novelty, interest and curiosity – have been developed by researchers in a number of domains [18]. This includes design science [11], developmental robotics [19] and computer games. Computational models of interest have been used for a number of purposes in game design. These include evaluating generated games [20], increasing the entertainment value of games [21] and controlling the behavior of NPCs in games [22]. In contrast, in this paper, Saunders’ [11] model of interest is used as part of a model of creativity. Saunders’ model is both flexible and modular, and thus lends itself well to adaptation to the PCG problem.

Saunders [11] proposed the model of interest in Eq. (1) as a model of creativity in design whereby artificial design agents can select design actions with the goal of generating moderately novel artifacts. Psychology literature suggests that there is an inverted U-shape relationship between novelty and interest [23,24]. That is, the most interesting experiences are often those that are moderately novel. Saunders [25] modeled interest I by first using a

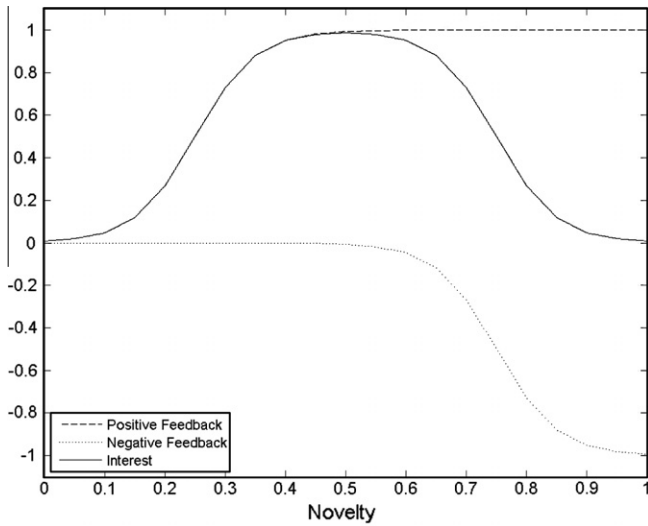


Fig. 2. The Wundt curve models interest in terms of positive and negative feedback for novelty. Parameter values used to generate this figure are $\rho^+ = \rho^- = 20$; $F_{\max}^+ = F_{\max}^- = 1$; $F_{\min}^+ = 0.25$; $F_{\min}^- = 0.75$.

real-time novelty detector [26] to compute the novelty N of a design, then applying the Wundt curve as follows:

$$I = F^+(N) - F^-(N) = \frac{F_{\max}^+}{1 + e^{-\rho^+(N - F_{\min}^+)}} - \frac{F_{\max}^-}{1 + e^{-\rho^-(N - F_{\min}^-)}} \quad (1)$$

The Wundt curve in Eq. (1) is the difference of two sigmoid feedback functions $F^+(N)$ and $F^-(N)$. $F^+(N)$ provides positive feedback for the discovery of novel designs while $F^-(N)$ provides negative feedback for highly novel designs. F_{\max}^+ is a constant defining the maximum positive feedback, F_{\max}^- defines the maximum negative feedback, ρ^+ and ρ^- are the slopes of the positive and negative feedback sigmoid functions, F_{\min}^+ is the minimum novelty to receive positive feedback and F_{\min}^- is the minimum novelty to receive negative feedback. The Wundt curve has a maximum value for moderate values of novelty (where novelty is in the range (0, 1)). This can be seen in Fig. 2, which plots interest against novelty.

Creativity is equated with interest using this model because the highest interest is allocated to designs that are similar-yet-different to existing designs. Similarity to existing designs means that aspects of usefulness and value can be retained, while novel or surprising variations can be introduced. In this paper the similarity of a design generated autonomously by a shape grammar to an existing design generated by a human is computed using the distance metric proposed in Section 3.3.

2.1.2. Shape grammars and design styles

A shape grammar [27,28] is a set of shape rules that can be applied to generate a set or language of designs. In general, applications of shape grammars have two main purposes. On one hand, shape grammars can be used as design tools to generate varieties of design languages. On the other hand, shape grammars as design analysis tools can be used both to analyze existing designs in order to better understand these designs, and to generate shape rules that produce the designs and other similar designs. This makes shape grammars a useful component of a system that will both decompose existing designs and create new designs.

A number of definitions have been proposed for shape grammars. This paper uses the following working definition of the basic components of a shape grammar as:

Σ : a finite set of shapes;

R : a finite set of shape rules;
 I : a finite set of initial shapes;
 F : a finite set of final shapes.

Each shape rule R takes the form $\sigma_{LHS} \rightarrow \sigma_{RHS}$ where $\sigma_{LHS} \in \Sigma$ and $\sigma_{RHS} \in \Sigma$ are two different labeled shapes. The shape rules of a grammar apply to an initial shape $I \in I$, and to designs produced by progressive application of rules to the chosen initial shape. Designs incorporating a final shape $F \in F$ are said to be in the language defined by a shape grammar.

Other shape grammar theories developed over the years include parametric grammars, color grammars, description grammars, structure grammars, parallel grammars, and so on, to address different aspects of designs. These, in turn, are relevant to different aspects of game design including color, texture and lighting, as well as building and place layout. The work in this paper adapts shape grammars to the layout of an instance game space and the placement of NPCs and objects within that space.

Shape grammars enable different designs that share a similar style to emerge by alternating the sequence of shape rule application. By analyzing existing designs of a known style, shape grammars can formally describe this known style and generate other designs that also share the style. Further, by incorporating additional devices, shape grammars are able to describe and generate new design languages in an extension to the original languages.

According to Knight [29], developing a shape grammar to describe and generate a known style involves the following steps:

- Defining a vocabulary of shapes and a set of spatial relations that are common to the design instances of the style;
- Defining shape rules that fix the occurrences of the spatial relations; and
- Providing an initial shape to start the application.

There are many successful design grammar examples for place designs in the physical world; for example, the Palladian grammar [30], the Mughul Gardens grammar [31] and the Prairie Houses grammar [32]. It thus follows that design grammars should be able to generate different level designs that capture unique and coherent styles.

One critical issue in the application of grammars for design is how to develop a grammar that produces designs which meet the design goals or constraints. Knight [33] suggests two different approaches that address this issue. The first approach is to incorporate foreknowledge into shape rules so that the generated designs could meet the given goals. To a certain extent, this approach is about constraining the grammars in order to control the shape rule application and increase the predictability of the outcomes. Knight [34] has analyzed a list of six different types of shape grammars ranging from unrestricted, standard shape grammars with the least predictable outcomes, to simple, restricted ones with the most predictable outcomes. The first approach requires adequate design knowledge while developing the shape grammar.

The second approach assumes inadequate design knowledge while developing the shape grammar. In the second approach, the shape grammar is allowed to generate designs without being constrained. After the designs are generated, an automated search and test device is then applied to search through the design space, test the designs and then select the ones that meet the given goals or satisfy the given constraints. The method in this paper is a hybrid of the two approaches. Some game space design knowledge is incorporated into the shape grammar and a computational model of creativity is used to select designs that are novel and surprising in the context of previously generated designs.

2.2. Procedural content generation

As a first step towards achieving PCG, it is necessary to understand the different roles of PCG as well as the elements of game content that go together in a game design. A range of taxonomies have been proposed for PCG and game content that achieve different purposes. These are discussed in the following subsections, which also place the work in this paper within the context of existing PCG literature.

2.2.1. Procedural content generation taxonomies

At a high level, Togelius et al. [1] draw a number of distinctions between different types of PCG. These include online versus offline PCG, necessary versus optional content, random seeds versus parameter vectors, stochastic versus deterministic generation and constructive versus generate-and-test versus search-based PCG. Other categories of PCG include experience-drive PCG [35–37] and related work on dynamic difficulty adjustment [38], which focus on game adaptation in response to learned cognitive models of player behavior.

Approaches that incorporate computational models of creativity fall loosely in the category of search-based PCG. That is, they progressively construct/generate and evaluate designs according to continuous valued fitness or utility functions. New candidate content is contingent upon the fitness value assigned to previously evaluated content. Cognitive models in this kind of PCG model the cognitive processes of game designers rather than game players.

Existing algorithmic technologies for search-based PCG include evolutionary systems, genetic algorithms, neural networks and shape grammars [1]. Creative systems are distinguished from search-based systems by that fact that the fitness function itself changes in response to designs previously experienced by the system. This implies that the experiences of the system will progressively influence what the system evaluates as a creative design. This is important to ensure that content is not merely ‘new’, but retains a connection with previous designs in terms of factors such as value or context. While creative systems may also have their foundations in evolutionary or grammar-based systems, they include mechanisms that modify the output of these algorithms subject to cognitively inspired models such as curiosity or situated reasoning.

PCG approaches also differ in the level of human input they use to generate designs. There is a spectrum of approaches that range from pseudo-random generators requiring minimal human input, to declarative modeling approaches [39] and design tools [40] that enable designers to state what they want to create in a virtual world. The work in this paper lies towards the middle of this spectrum. Specifically, our framework requires one or more initial human designs which are then decomposed into basic elements of game content. These elements are then recombined autonomously by the creative PCG system to generate new designs. The next section reviews existing taxonomies of the elements of game content.

2.2.2. Game content taxonomies

A number of taxonomies have been proposed for the elements of game content that are combined to comprise a game design. For adventure games, for example, Dormans [41] divides the content of a level into two categories: mission and space. The mission category includes more specific content types describing who players will meet (NPCs), what players will find (interactive objects) and what players will do (puzzles, storyline, plot elements). The space category describes where a player is at a point in a mission. A mapping is assumed between the mission and space categories such that content in the space supports content in the mission.

For example, if there is a locked door object in the mission there must be a doorway in the space to locate the door.

Hendrikx et al. [2] propose a more detailed taxonomy to cover a wider range of game genres. Their taxonomy has a hierarchical structure whereby content elements at lower levels in the hierarchy combine to create content at higher levels. The lowest level content category comprises ‘game bits’ or elementary units of game content. Hendrikx et al. include textures, sounds, vegetation, buildings, behavior, fire, water, stone and clouds as examples of game bits. Higher level categories are proposed for the game space, game systems, game scenarios, game design and derived content. The game space category includes subcategories for indoor and outdoor maps; game systems include subcategories such as eco-systems, road networks and NPC behavior; and game scenarios include subcategories such as puzzles and stories.

Other genre-independent taxonomies of game content [42,43] identify similar elements of game content including levels, geometry, textures, props, lighting, sound, mood, NPCs, plots, quests, and user interface. These taxonomies consider game elements in parallel rather than as a strict hierarchy.

While in future it may be possible for PCG systems to generate all kinds of game content, existing work tends to focus on a subset of game content such as indoor or outdoor map generation, ecosystems or story generation. In practice these focused systems select a subset of a full game content taxonomy to use as the basis for PCG. The work in this paper takes the same approach and focuses on PCG of instances for MMORPGs. This type of game is reviewed briefly in the next section.

2.3. Massively multiplayer, online role-playing games

Massively multiplayer, online role-playing games (MMORPGs) are a genre of computer games characterized by a very large number of players interacting with each other in a persistent virtual game world. Players assume the role of a character in keeping with the theme of the game world. They interact with the game world, and with each other, by controlling their character's actions. The primary goal in most MMORPGs is for players to develop and ‘grow’ their character by completing quests to earn experience points.

A quest can be thought of as a mission or task to be completed by a player-controlled character or group of characters to gain a reward [44]. Quests may be presented to players through in game objects such as notice-boards, or by NPCs. These quests direct the player to go to a specified location and carry out the specified task. Thus, as with the mission-space mapping proposed by Dormans [41] for adventure games, there is a quest-space mapping for MMORPGs. Existing work has studied quest design for MMORPGs in detail [44,45]. The work in this paper focuses on space design for quests, assuming that there are key elements of game content around which certain kinds of quests are designed.

Quests can be grouped in four broad categories: combat-quests, gather-quests, fetch/deliver-quests and escort-quests. In MMORPGs, there is a super-class of quest that, rather than occurring in the persistent game world, take the player (or group of players) into a special type of location called an ‘instance’. Instances (short for instance dungeons) are sub-areas where players are able to interact privately with the virtual game world. Players are temporarily transported to a private copy of the specified sub-area and can interact with that part of the game without interference from other players. This reduces competition, while also reducing the amount of data that needs to be sent to and from the server. Instances often feature rarer or more difficult enemies, as well as rarer or more valuable reward items.

The self-contained nature of instances, the ability to separate quest and space, and the pressure for continuous development of

Table 1

A high level game content taxonomy for MMORPG instance spaces.

Symbol	Category
<i>S</i>	Terrain
<i>O</i>	Object
<i>N</i>	Non-player character
<i>P</i>	Plot fragment
<i>Q</i>	Quest

Table 2

Expansion of two categories of the game content taxonomy for MMORPG instance spaces: NPC and object.

Symbol	Category	Subcategories
	Object	Usable
O_1		Furniture
O_2		Artifact
		Portable
O_3		Ammunition
O_4		Treasure
O_5		Natural resource
		Consumable
O_6		Food
O_7		Drink
		Wearable
O_8		Clothing
O_9		Armor
O_{10}		Weapon
	Non-player character	Enemy
N_1		Rare static
N_2		Rare mobile
N_3		Common static
N_4		Common mobile
		Partner character
N_5		Companion/pet
N_6		Mount
N_7		Support character

MMORPGs to provide players with new avenues for character development makes them an ideal candidate for PCG. MMORPG instances (specifically instance spaces for combat quests) are thus used as the focus for studies and examples in the remainder of this paper.

3. Combining computational creativity and generative design for MMORPG instances

Fig. 1 illustrated the high level framework for computational creativity in PCG proposed in this paper. First, to permit new designs to capture aspects of the usefulness and value of high quality

human designs, our framework assumes the existence of at least one initial human design as a starting point for the creative process. This design is then decomposed (Step I) into primitive components that become the design variables input to the creative process (Step II). The creative process recombines those components to create new designs. This paper presents a tool that aids the human designer in Step I and automates the role of the human in Step II. Future work will also consider the automation of Step I. This is considered further in Section 5.

Our framework assumes mechanisms for both describing existing designs and generating new designs. The shape grammar formalism provides us with the foundation for such a mechanism. The remainder of this section is concerned with establishing the formalisms to describe our framework. Section 3.1 proposes a game content taxonomy for decomposing MMORPG instances. Section 3.2 uses the elements of this taxonomy to propose a shape grammar formalism for generating new game instances. Section 3.3 describes a method for incorporating a computational model of interest to evaluate generated designs. A demonstration and evaluation of a tool that implements our framework is described in Section 4.

3.1. A game content taxonomy for MMORPG combat instances

Combat quest instances are a good candidate for PCG because they follow a well-defined structure. This permits the space to be generated somewhat independently of the quest, which may itself be generated by PCG or by a human designer. Generation of the quest is not considered further in this paper. Rather, we assume the following generic definition of a combat quest for an instance:

A player (or group of players) must kill some quota of (possibly different kinds of) enemies or monsters within the instance area.

Table 1 proposes a high level taxonomy of game elements for instance spaces. Five broad categories are proposed for the space, objects, NPCs, plot fragments and quests.

According to the taxonomy proposed by Hendriks et al. [2] it is assumed that elements in these categories may be comprised of elements from other lower level categories. For example, an object may be comprised of textures from the ‘game bits’ category. More importantly, in this paper, each of the five categories listed in Table 1 may be broken into sub-categories relevant to a particular game. Table 2 provides a sample expansion of the object and NPC categories. Section 3.2 describes how we can use these sub-categories, and define a terrain category, as the basis for a shape grammar for combat instance design.

This game content taxonomy is necessarily simple in that it defines a relatively small set of symbols and sub-symbols for content categories and sub-categories. However, by associating specific

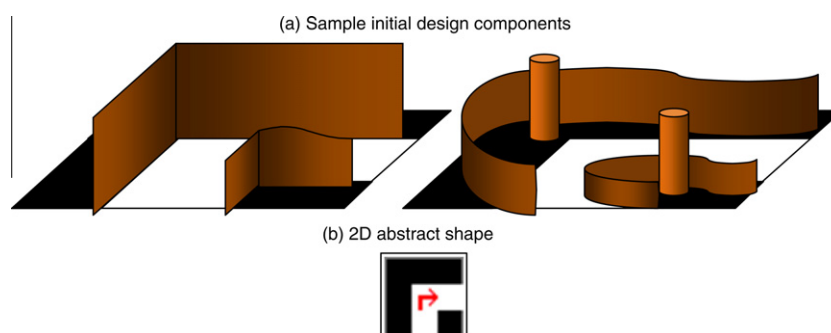
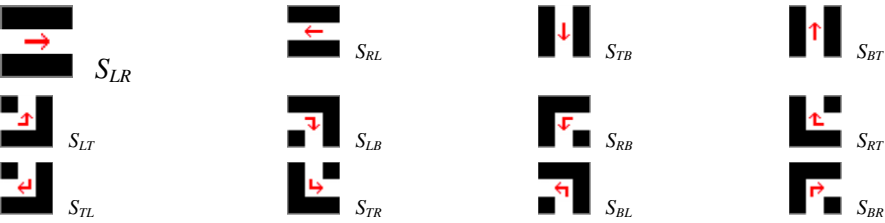


Fig. 3. A 2D abstract representation can be used to represent different initial design components. (a) Two different initial designs; (b) A 2D grid square representation.

Table 3
Terrain shapes ($S \in \Sigma$).

Type of shape	Shapes
Basic terrain	

content types with a symbolic notation, it also has a level of precision that has not been considered in broader game content taxonomies. This level of precision is necessary to automate content generation of (parts of) these taxonomies.

3.2. A shape grammar for instance design for combat quests

In this paper, the set Σ of shapes comprises four subsets for basic terrain shapes **S**, compound terrain shapes **C**, shapes to locate NPCs **N** and shapes to locate objects **O**. That is:

$$\Sigma = \{\mathbf{S} \cup \mathbf{C} \cup \mathbf{N} \cup \mathbf{O}\} \quad (2)$$

The following paragraphs consider shapes in each of these categories in turn.

S and **C** are sub-categories of the terrain category in Table 1. In fact, various sub-categories could be chosen for the terrain category. For example, a distinction between indoor and outdoor terrains could be made. For indoor terrains, a category of constraining shapes might include walls, floors and roofs. A category of non-constraining shapes might include doors, windows or trapdoors and so on. Dormans [41] uses symbols for walls, open-spaces and connections to generate space. However, in this paper, we use slightly more complex primitive space elements based on a set of abstract shapes that fit within a 2D square grid. In this paper we consider the 2D grid as an abstraction that could represent a more complex (possibly even 3D space), depending on the nature of the initial design and how it is decomposed. Two examples are shown in Fig. 3. Fig. 3(a) shows two different initial design components that can be decomposed to the same abstract shape within a 2D grid in Fig. 3(b). A grammar-based PCG system can work with the abstract shape, then construct a new design by overlaying the original components.

More formally, in this paper, basic terrain shapes S represents a square region of the terrain and have an entry and an exit point. The entry and exit points are on two of the four sides of the 2D square. Each shape is named according to the convention S_{PQ} . The value of the labels P and Q correspond to the entry and exit points for the shape. The possible values of P and Q are L (left), R (right), T (top), and B (bottom) assuming a plan view. A list of basic terrain shapes is given in Table 3. It should be noted that no assumption is made as to what happens between the entry and exit points. Rather, it is the decomposition of the initial design that determines the components that will be overlaid on the abstract shapes (see Fig. 3 and the demonstrations in Section 4).

To capture specific design styles characteristic of a particular game instance, it can be advantageous to permit more complex customized terrain shapes that combine some of the basic terrain shapes in Table 3. These shapes are referred to as compound-shapes. An example of a compound shape is shown in Fig. 4. Compound-shapes are named according to the convention C_i where i is a unique numeric identifier $i = 1, 2, 3 \dots$

In addition to basic and compound terrain shapes, Eq. (2) includes shapes that locate NPCs and objects. Abstract representations for such shapes are shown in Table 4. These shapes are named according to the conventions: N_{jPQ} for NPCs and O_{kPQ} for objects. $j = 1 \dots 7$ and $k = 1 \dots 10$ are identifiers for the type of NPC or object respectively, as defined in Table 2. The label P identifies the location of the object or NPC within the grid square. The label Q identifies the type of NPC or object according to the taxonomy in Table 2. For example in an instance that supports a combat quest Q in the range [1,4] may be appropriate.

Finally, initial shapes I that represent the start of a level are labeled using a similar convention to terrain shapes. However, rather than having an entry point, the start point is placed in the middle of the tile, denoted C (center). Likewise, final shapes F that represent the end of the level use the C label in place of the exit point. Table 5 shows possible initial and final shapes, assuming a 2D grid layout.

To decompose a design, a shape grammar needs to include basic terrain shapes and compound-shapes that capture the style of the design. In addition it needs to include shape rules that capture how these shapes can be combined. Shape rules may describe the possible basic terrain shapes that can be concatenated from a given initial shape, or the way that NPCs or objects can be overlaid on a terrain shape. Table 6 gives the notation for a number of sample shape rules and illustrates their effect.

Once a design has been decomposed into basic terrain shapes, compound-shapes, NPC shapes, object shapes, compound-shapes and rules, the rules can be applied randomly or probabilistically to generate new designs. A new design starts with an initial shape and is finished when a rule fires that adds a final shape to the design. A generated abstract design comprising three types of basic terrain shape, two types of compound-shape, an initial shape, a final shape and an object shape is shown in Fig. 5. The abstract shapes can now be overlaid with the real components from which the grammar was built.

Formally, a new design D is a sequence of shapes $\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_t, \dots, \sigma_T$ generated over a sequence of time-steps $t = 1 \dots T$. One rule fires at each time-step t , adding one shape σ_t to the design. In random generation, the rule to fire is selected at random from the set of valid rules at each time-step. The set of valid rules

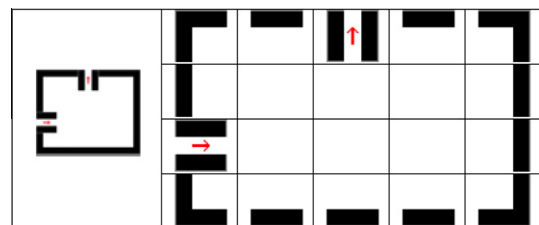


Fig. 4. A sample compound-shape.

Table 4Shapes for locating NPCs and objects ($N \in \Sigma$ and $O \in \Sigma$).





Type of shape	Shapes			
NPC		N_{jLQ}		N_{jRQ}
Object		O_{kLQ}		O_{kRQ}

Table 5

Initial and final shapes (I and F) for an instance layout grammar.









Type of shape	Shapes			
Initial		I_{CR}		I_{CL}
Final		F_{LC}		F_{RC}

Table 6

Sample shape rules and visualizations of their effects.

Shape rule	Left Hand Side (Before rule application)	Right Hand Side (After rule application)
$I_{CR} \rightarrow I_{CR} + S_{LR}$		
$S_{LR} \rightarrow S_{LR} + N_{RQ}$		

$\mathbf{R}'_t \subseteq \mathbf{R}$ at time t comprises all rules with the most recently added initial shape or terrain shape on the left hand side. That is:

$$\mathbf{R}'_t = \{\sigma_{LHS} \rightarrow \sigma_{RHS} \mid \sigma_{LHS} = \sigma_{t-\tau} \wedge \tau = \min_{1 \leq \tau} (\sigma_{t-\tau} \in \mathbf{I} \cup \mathbf{S})\}$$

This represents an unconstrained approach to generating levels.

In contrast, probabilistic generation represents a more restricted approach to generating levels. This paper proposes an approach to probabilistic generation that permits the designer to specify desired probabilities for different types of shapes. At each time-step t there is a desired probability $P_t(\sigma)$ of each shape σ occurring in the design. The initial desired probability of each shape $P_0(\sigma)$ is specified by the human designer. The actual frequency of each shape σ in the design at time-step t is:

$$P'_t(\sigma) = \frac{\text{count}_t(\sigma)}{t} \text{ where } \text{count}_t(\sigma) = \sum_{\tau=1}^t (\sigma_\tau = \sigma)$$

At each time-step, the system compares the initial desired and current actual frequencies and computes a weight for each shape. The weight indicates whether that shape should be further favored or ignored at the next time-step in order to achieve the initial desired probabilities.

$$W_{t+1}(\sigma) = \frac{(P_0(\sigma))^2}{P'_t(\sigma)}$$

Adjusted desired probabilities are then computed for each unique shape in the design by normalizing across all desired weights as follows:

$$P_{t+1}(\sigma) = \frac{W_{t+1}(\sigma)}{\sum_{\sigma' \in D} W_{t+1}(\sigma')}$$

Adjusted desired shape probabilities are used to determine the probability with which each valid rule will fire at a given time-step. Valid rules are divided into subsets \mathbf{R}_σ such that:

$$\mathbf{R}_\sigma = \{\sigma_{LHS} \rightarrow \sigma_{RHS} \mid \sigma_{LHS} = \sigma\}$$

The probability of one of the rules $R_\sigma \in \mathbf{R}_\sigma$ firing is:

$$P(R)_\sigma = \frac{P_{t+1}(\sigma)}{|\mathbf{R}_\sigma|}$$

Terrain shapes, NPC and object shapes can be considered as a single set, or separately using this approach.

3.3. Integrating a computational model of interest with a shape grammar

The shape grammar formalism permits design knowledge to be incorporated into the generated design. Using a shape grammar in conjunction with a computational model of creativity can permit a subset of generated designs to be selected that are novel and surprising while maintaining their value in the context of the original design style. One model of creativity that has been used in design is a computational model of interest [11]. In this model, designs are evaluated in the context of other designs previously experienced by the designer. We adapt this model to the design of computer game levels here.

First, the design is parameterized to give a vector representation $D = (d^1, d^2, d^3, \dots, d^k, \dots, d^x)$ that permits comparison of designs. Each parameter d^x corresponds to a group of shapes of the grammar such that NPC and object shapes are grouped with the basic, compound, initial or final shape to which they apply. NPC shapes $N \in \mathbf{N}$ and object shapes $O \in \mathbf{O}$ apply to the last preceding basic, compound, initial or final shape. Given two parameterized designs D_r (a previously experienced reference design comprising X shapes) and D_t (a new test design comprising Y shapes), a distance metric is used to compute the overall difference between the two vectors D_r and D_t . The resulting distance value can be thought of as representing the novelty N of the new design.

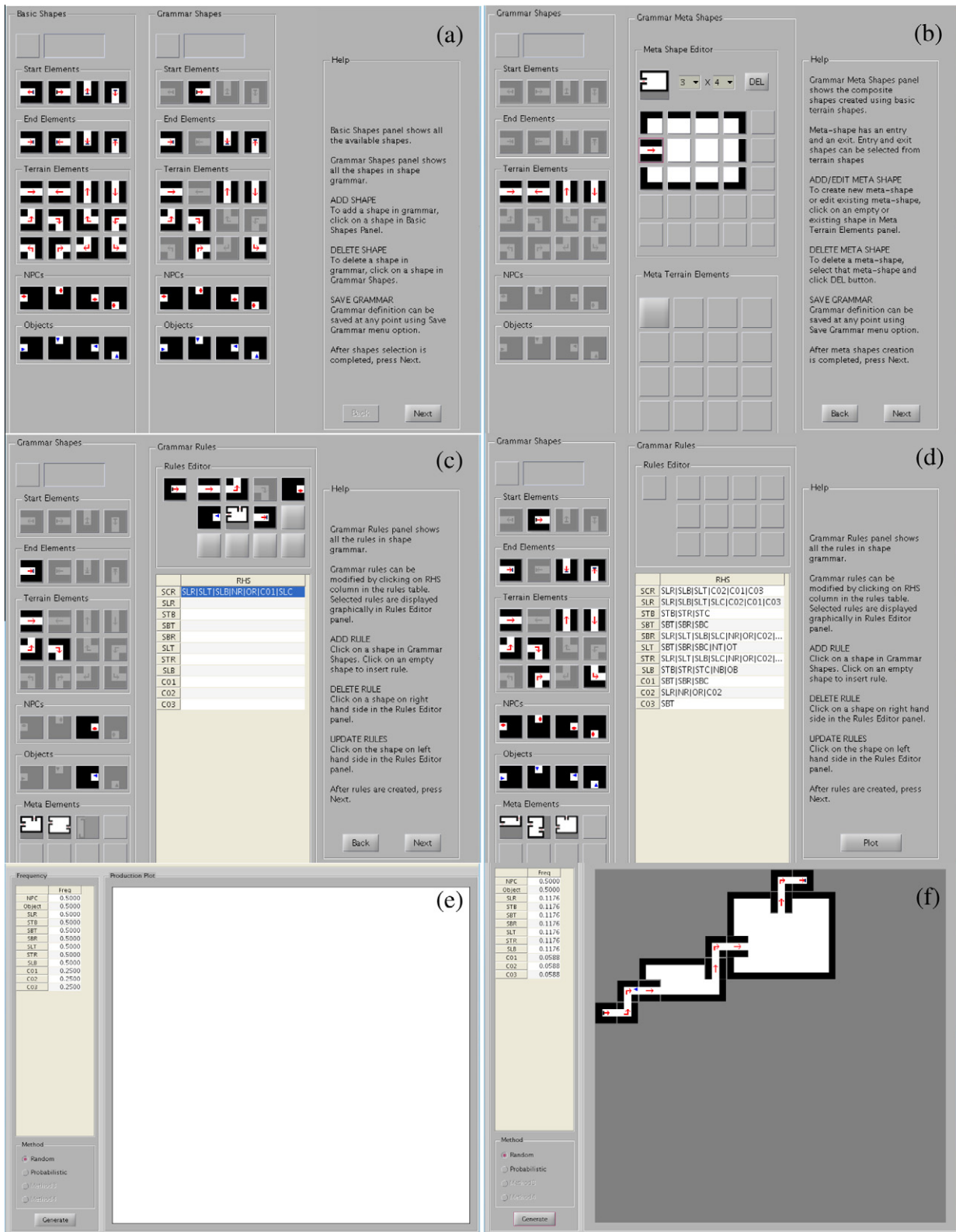


Fig. 5. The Shape Grammar Designer application that implements our framework for creativity in PCG. (a) Step 1: selecting start, end, NPC and object shapes. (b) Step 2: creating compound-shapes. (c) Step 3: defining shape rules. (d) A completed grammar, ready to generate and "Plot" new designs. (e) Setting relative shape frequencies and selecting the generation type. (f) A generated design.

In this paper we consider a model of novelty based on the difference between designs experienced by the generative system. We propose a distance measure that accommodates different length vector parameterizations of designs, as well as micro-varia-

tions in designs as part of its overall, holistic measure of difference between the two designs. A form of dynamic programming [46,47] was employed, as defined by Eqs. (3)–(11) below. In essence this approach seeks an optimal alignment of two designs that mini-

mizes their overall difference. Each design is allowed to be 'warped' within certain constraints by limited repetition of shape groups d^x . A warp-path W is then computed using Eqs. (3) and (4) that travels from the beginning of both levels (d_r^1 and d_t^1) to their ends (d_r^X and d_t^Y), while ensuring optimal alignment is achieved and that all shape groups d_r^x and d_t^y within the two designs contribute to that alignment. No shape group can be skipped and shape groups must occur in the order defined by the level designs D_r and D_t .

The overall distance $\|D_r, D_t\|$ between two designs D_r and D_t is measured as the accumulated distance $\Delta(X, Y)$ when the last two shape groups d_r^x and d_t^y of both designs are incorporated in a warp-path. We denote this:

$$\|D_r, D_t\| = \Delta(d_r^x, d_t^y) \quad (3)$$

$\Delta(d_r^x, d_t^y)$ can be computed recursively. The cumulative distance $\Delta(d_r^x, d_t^y)$ where $x \in \{1 \dots X\}$ and $y \in \{1 \dots Y\}$ is the sum of the difference between the current shape groups from the two levels designs $\delta(d_r^x, d_t^y)$ plus the minimum partial warp-path-distance up to that point:

$$\Delta(d_r^x, d_t^y) = \delta(d_r^x, d_t^y) + \min \left\{ \begin{array}{l} \Delta(d_r^{x-1}, d_t^y) \\ \Delta(d_r^x, d_t^{y-1}) \\ \Delta(d_r^{x-1}, d_t^{y-1}) \end{array} \right\} \quad \text{if } x, y \neq 1 \quad (4)$$

or

$$\Delta(x, y) = \delta(d_r^1, d_t^1)$$

The warp-path W comprises shape groups d_r^x and d_t^y in the order that they were visited during the recursive definition of $\|D_r, D_t\|$.

The difference $\delta(d_r^x, d_t^y)$ between two shape groups d_r^x and d_t^y is the sum of four individual difference functions as follows:

$$\delta(d_r^x, d_t^y) = T_\delta(d_r^x, d_t^y) + E_\delta(d_r^x, d_t^y) + N_\delta(d_r^x, d_t^y) + O_\delta(d_r^x, d_t^y) \quad (5)$$

$T_\delta(d_r^x, d_t^y)$ is a shape type difference defined as follows:

$$T_\delta(d_r^x, d_t^y) = \begin{cases} 4 & \text{type}(d_r^x) \neq \text{type}(d_t^y) \\ 0 & \text{type}(d_r^x) = \text{type}(d_t^y) \end{cases} \quad (6)$$

Shape types are defined according to Tables 3 and 5.

$E_\delta(d_r^x, d_t^y)$ is an entrance/exit difference defined in Eq. (7). $E_{\text{entrance}}(d_r^x, d_t^y)$ is a count of the number of sides difference between the position of the exit on shapes d_r^x and d_t^y . For instance if d_r^x has its entrance at the top (T) and d_t^y has its entrance at the right (R) then there is a one side difference and $E_{\text{entrance}}(d_r^x, d_t^y) = 1$.

$$E_\delta(d_r^x, d_t^y) = E_{\text{entrance}}(d_r^x, d_t^y) + E_{\text{exit}}(d_r^x, d_t^y) \quad (7)$$

$N_\delta(d_r^x, d_t^y)$ is an NPC difference defined in Eqs. (8) and (9), which accounts for differences in the number $n_{\text{NPC}}(\cdot)$ and location $l_{\text{NPC}}(\cdot)$ of NPCs

$$N_\delta(d_r^x, d_t^y) = \sum_{\text{side}=\text{T,B,L,R}} |n_{\text{NPC}}^{\text{side}}(d_r^x) - n_{\text{NPC}}^{\text{side}}(d_t^y)| + l_{\text{NPC}}(d_r^x, d_t^y) \quad (8)$$

where

$$l_{\text{NPC}}(d_r^x, d_t^y) = \begin{cases} 4 & \text{if } \sum_{\text{side}=\text{T,B,L,R}} n_{\text{NPC}}^{\text{side}}(d_r^x) \neq \sum_{\text{side}=\text{T,B,L,R}} n_{\text{NPC}}^{\text{side}}(d_t^y) \\ 0 & \text{if } \sum_{\text{side}=\text{T,B,L,R}} n_{\text{NPC}}^{\text{side}}(d_r^x) = \sum_{\text{side}=\text{T,B,L,R}} n_{\text{NPC}}^{\text{side}}(d_t^y) \end{cases} \quad (9)$$

$O_\delta(d_r^x, d_t^y)$ is an object difference defined in Eqs. (10) and (11). This is computed similarly to the NPC difference

$$O_\delta(d_r^x, d_t^y) = \sum_{\text{side}=\text{T,B,L,R}} |n_{\text{object}}^{\text{side}}(d_r^x) - n_{\text{object}}^{\text{side}}(d_t^y)| + l_{\text{object}}(d_r^x, d_t^y) \quad (10)$$

where

$$l_{\text{object}}(d_r^x, d_t^y) = \begin{cases} 4 & \text{if } \sum_{\text{side}=\text{T,B,L,R}} n_{\text{object}}^{\text{side}}(d_r^x) \neq \sum_{\text{side}=\text{T,B,L,R}} n_{\text{object}}^{\text{side}}(d_t^y) \\ 0 & \text{if } \sum_{\text{side}=\text{T,B,L,R}} n_{\text{object}}^{\text{side}}(d_r^x) = \sum_{\text{side}=\text{T,B,L,R}} n_{\text{object}}^{\text{side}}(d_t^y) \end{cases} \quad (11)$$

The design of the local shape group difference as defined by Eqs. (5)–(11) drives the dynamic alignment; as it is the series of individual, sequential shape group comparisons, and the accumulation of those scores, that is minimized. The particular construction of Eq. (5) as the sum of a type difference, entrance/exit difference, NPC difference, and object difference – and in particular their relative weights – is arbitrary in that alternate schemes and weightings can easily be constructed. However, it is argued that this scheme gives relatively even weighting to topological and spatial considerations (the direction of flow as defined by the entrance/exit score and the type match scores) as well as to NPC and object scores. It could also be extended to include other kinds of game content, defined by a more complex game content taxonomy.

It should be noted that, aside from the representation of a design in terms of the components of a game content taxonomy, the distance measure does not attempt to incorporate design knowledge. Penalty values of 0–4 are given for differences in a particular type of content, but no attempt is made to distinguish between or equate differences across content categories. Rather, it is the role of the shape grammar to incorporate design knowledge. The distance-based novelty metric is then a relatively generic cognitive model that distinguishes between a set of designs with otherwise similar properties.

While the overall distance $\|D_r, D_t\|$ when both designs are aligned is the most obvious candidate for a difference measure, other candidates exist, derived from the warp-path W and the length $|W|$ of the warp-path. In other domains these have been shown to encode useful information concerning the relative difference of two vectors [48]. In this paper we consider three aspects of difference:

- *dist*: a normalized distance measure where $\|D_r, D_t\|$ is normalized with respect to the maximum possible distance between any paths of length X and Y :

$$\frac{\|D_r, D_t\|}{\max(X, Y)}$$

This is a measure of the overall cumulative difference between the two designs – a holistic measure of difference.

- *misalign*: is the normalized difference between the warp-path length $|W|$ and the minimum possible warp-path length for two identical designs:

$$\frac{|W|}{\max(X, Y)}$$

This is a measure of how much additional warping (repetition) of shapes in one or the other design was needed to achieve optimal alignment. It is a measure of macro-structural difference.

- *match*: the fraction of shape groups in the two designs that are exactly the same once the designs have been aligned. If we consider the warp-path W as a vector of tuples being monotonically increasing indexes $\{(w_{r1}, w_{t1}), (w_{r2}, w_{t2}), \dots, (w_{r|W|}, w_{t|W|})\}$ into the two designs D_r and D_t then match is defined as:

$$\text{match} = \frac{\text{count}_i \delta(d_r^{w_{ri}}, d_t^{w_{ti}}) = 0}{|W|}$$

This is a measure of the fraction of the two designs that are exactly the same after they have been aligned. That is, the fraction of the two designs possessing the same shapes, entrance and exit loca-

tions, and numbers and locations of objects and NPCs at the same positions within their designs.

We consider difference as $(1 - match)$ and compute the novelty of the design using the average of $(1 - match)$, $dist$ and $misalign$:

$$N + 1/3[dist + misalign + (1 - match)]$$

A design is considered interesting – and thus creative – if it is ‘similar-yet-different’ to previously experienced designs. In other words, a design is considered interesting and creative if there is a ‘moderate difference’ identified by applying the Wundt curve from Eq. (1). A design that is too similar to existing designs is not interesting or creative because it lacks originality. A design that is too different to existing designs will not be interesting because it does not fit the context of the original design. The most creative designs using this model are moderately different to (one or more) previously experienced designs. This means that they potentially incorporate aspects of usefulness and value from the original design with novel or surprising variations.

4. Demonstration and evaluation

Section 4.1 describes a system that implements the framework for creative design generation described in Section 3. Section 4.2 demonstrates an example of the system in use and evaluates the creativity of generated designs.

4.1. Shape Grammar Designer: a system for aiding creative design

Shape Grammar Designer is a tool that (I) aids a designer to decompose an original design and (II) uses the elements of the decomposed design to create new designs. The current version of *Shape Grammar Designer* implements only a subset of the functionality described in Section 3, sufficient to demonstrate the various types of shapes in Table 1 (basic shapes, compound-shapes, NPCs and Objects). The current version does not yet distinguish between all of the sub-categories in Table 2.

4.1.1. Decomposing a Human Design

Decomposing a design requires three steps that roughly correspond to those suggested by Knight [29] for developing a shape grammar to describe a known design style. Each of these steps is supported by the *Shape Grammar Designer*:

1. Select basic shapes in the grammar, including start and end shapes;
2. Create compound-shapes; and
3. Specify grammar rules.

Step 1 is shown in Fig. 5(a), which shows the three panel screen layout of the application. The panel on the left hand side, “Basic Shapes”, lists all possible basic terrain shapes, including initial and final shapes. The middle panel, “Grammar Shapes” shows the shapes selected so far in the grammar decomposition of the original design. The rightmost “Help” panel provides instructions to the user. The current interface permits only one type of NPC and one type of object, but this could be extended in future to permit different types of NPCs and objects as per the taxonomy in Table 2.

Once initial, final, terrain, NPC and object shapes have been selected, the next step is activated by clicking “Next” button. The next step is the creation of compound-shapes. In Step 2, three panels are displayed as shown in Fig. 5(b). The leftmost panel is “Grammar Shapes” showing possible basic terrain shapes. The next panel “Grammar Meta Shapes” is the tool to design compound-shapes. There are two areas in “Grammar Meta Shapes” panel.

The top area “Meta Shape Editor” provides a space to create a compound-shape of maximum size 5×5 . The area “Meta Terrain Elements” lists all defined compound-shapes created so far.

To create a new compound-shape, the user clicks on the empty shape in “Meta Terrain Elements” area. This enables “Meta Shape Editor” panel. The user then selects the size of the compound-shape using the drop-down lists. Based on the number of rows and columns entered, an empty compound-shape with borders is displayed. Four terrain shapes are enabled in the “Grammar Shapes” panel to permit an entry and exit to be marked on the compound-shape. To mark an entry, a terrain shape is selected. The selected shape is displayed in the top square of the “Grammar Shapes” panel. Next the user clicks on one of the compound-shape tiles in the “Meta Shapes Editor” panel to locate the entry. An exit point can be marked in a similar manner. Once an entry and exit are marked, the compound-shape can be saved by clicking on the square button (next to size pull-downs) displaying a mini version of the new compound-shape. The shape is then saved and is displayed in the “Meta Terrain Elements” panel. The process can be repeated to create further compound-shapes. In this paper, for simplicity, compound-shapes are rectangular and have one entry and one exit point. More complex shapes with multiple entry and exit points could be considered in future.

Step 3 is the specification of grammar rules. This is accessed via the “Next” button. In Step 3, three panels are displayed as shown in Fig. 5(c). The leftmost panel “Grammar Shapes” shows the basic terrain shapes and the compound-shapes created previously. The “Grammar Rules” panel allows users to specify shape rules using a visual interface. Each rule is of the form $\sigma_{LHS} \rightarrow \sigma_{RHS}$. The table in “Grammar Rules” panel lists all the defined rules using the notation in Tables 3–5. To add a new rule or edit existing rules, the user clicks on the RHS column corresponding to the desired σ_{LHS} . This enables the “Rules Editor” panel and shows σ_{LHS} on the leftmost button in the “Rules Editor” panel (see Fig. 5(c)).

In addition, all the shapes that could follow σ_{LHS} are highlighted in the “Grammar Shapes” panel. For example, for the start shape S_{CR} , all the shapes that have an entry on the left edge could follow, and any of those shapes can be selected to create rules. To add a σ_{RHS} in a rule, the user can click on any of the highlighted shapes in the “Grammar Shapes” panel. The selected shape is displayed in the top square in the “Grammar Shapes” panel. The user then clicks on an empty button on the right hand side of the “Rules Editor” panel. For example, if the S_{LR} shape is selected, this creates a rule $S_{CR} \rightarrow S_{LR}$. More rules can be added by selecting a shape from “Grammar Shapes” and assigning it to an empty button in the “Rules Editor”. Once all the rules have been defined for a given σ_{LHS} , they can be saved by clicking on the σ_{LHS} button. This updates the rules table. To edit an existing rule, the user can click on the rule in the table. This populates the “Rules Editor” with the appropriate σ_{LHS} and various σ_{RHS} . New rules can be added by selecting shapes and assigning them to empty squares on right hand side. Existing rules can be deleted by clicking on them.

At any point in the grammar specification process, the current grammar definition can be saved using the “Save Grammar” menu item under the “File” menu. The user can go back to previous steps using “Back” button and any rules saved in the table are retained. In case there is a change in the basic shapes or compound-shapes, the saved rules are automatically updated to exclude any deleted shapes.

This is the last step of the design decomposition process. The next step uses the grammar to create new designs automatically. This occurs in a separate window. The user opens their grammar in a new window and the “Plot” button is now active as shown in Fig. 5(d). The shapes and the rules are displayed on the screen. The user clicks the “Plot” button to open a “Plot” window as shown in Fig. 5(e).

The “Method” panel is used to select the method to generate designs. Currently there are two methods: Random and Probabilistic. In the random approach, a design generated over a sequence of time-steps t , with one rule firing at each time-step, adding one shape to the design. Depending on the previous shape added to the design a subset of rules R' will be valid. The random method selects randomly from valid rules to create a design.

The probabilistic method uses the frequency table in the “Frequency” panel to try to ensure that the frequency with which each shape occurs in the new design matches user specified values, as described in Section 3.2.

Once a method is selected (and initial desired probabilities set if required), the user clicks on the “Generate” button. The system uses the selected method to create a new design and displays it in the “Production Plot” area as shown in Fig. 5(f). Every time the “Generate” button is clicked, a new design is created. An example is shown in

The “File” menu can be used to save the parameters corresponding to the current design. The parameters saved in a file include frequency values or the random seed used. These parameters can be retrieved later to recreate the current design. The generated design can also be saved as an image using the “Save Figure” menu item.

4.2. Demonstration and evaluation

This section shows how existing designs can be decomposed and the decomposition (in the form of a shape grammar) used to create new designs that are creative adaptations of the original. Creativity is evaluated quantitatively in terms of interest using the model described in Section 3.

4.2.1. The Scarlet Monastery Armory (World of Warcraft)

In this demonstration, the initial design is a 2D plan view of Scarlet Monastery Armory from the game *World of Warcraft* [49]. Fig. 6 shows the rough idea for a possible decomposition of the armory. Fig. 6(a) shows the location of NPCs, while Fig. 6(b) gives a rough grid-based decomposition. Fig. 7 shows a shape grammar decomposition of the Scarlet Monastery armory using the *Shape Grammar Designer*. Fig. 8 shows that this grammar can produce a design that approximates the original design.

Fig. 8 shows twelve system-generated designs, generated by successive use of the ‘Generate’ button as described in Section 4.1. The designs exhibit different levels of computational creativity according to the model defined in Section 3. The raw difference values between the generated levels and the original *Scarlet Monastery Armory* in Fig. 7 according to the *dist*, *misalign* and inverse match ($1 - match$) metrics are plotted in Fig. 10. Fig. 10 shows that there is some correlation between the metrics, but there are also differences that are a result of each metric comparing designs in a different way. The average (novelty) value and creativity values for each design are shown in Fig. 10. We use the interest parameters in Fig. 2 to compute creativity.

The first result apparent from Fig. 9 and Fig. 10 is that the novelty and interest values for all designs fall in a similar range; that is, ($0.2 < N < 0.5$) and ($0.7 < I < 1$) respectively. This is justified because all designs are generated by the same human specified design grammar. The following paragraphs discuss the creativity of the generated levels with reference to the composition of the game content in the levels. We use the term ‘creativity’ here because our approach incorporates two broad aspects of creativity. First, usefulness and value are captured in a new design by ensuring that there is similarity or ‘closeness’ between new designs an existing human design. ‘Closeness’ in this paper is computed by the distance measure described in Section 3. Novel and surprising varia-

tions are introduced in new designs by selecting those that are a moderate distance from the original design.

The game levels are considered in three brackets corresponding to different interest levels (which in turn take into account distance, misalignment and mismatch to the original human design). The three brackets are: very high creativity ($0.90 \leq I \leq 1$); high creativity ($0.8 \leq I < 0.9$) and moderate creativity ($I < 0.8$).

Designs A, C, D, F, G, H, J and K fall in the first bracket. Inspection of these designs in Fig. 8 shows that these level designs do indeed capture some of the important aspects of the original *Scarlet Monastery Armory*, but with some ‘twist’.

- Level A ($I = 0.95$) comprises both types of compound shape and a variety of basic shapes interspersed with NPCs. In addition, it ends with an NPC to fight. However it is quite a bit shorter than the original, resulting in a moderate novelty value and relatively high computational interest.
- Level C ($I = 0.98$) also comprises both types of compound shape and a variety of basic shapes interspersed with NPCs. In contrast to the original the level ends with an object to gather in a courtyard and unique zig-zag hallway. These differences result in a moderate novelty value and a very high interest value.
- Level D ($I = 0.91$) is similar to level A in that it is a relatively short combination of basic and compound shapes. It also includes one of the characteristic U shaped bends of the original level. This gives it a lower novelty value than level A. It is interesting to note that level D doesn’t end with an NPC to fight, which could make it somewhat anticlimactic as a game level. It thus seems correctly classified at a lower interest than level A as it doesn’t include one of the key cultural aspects of a combat instance.
- Level F ($I = 0.94$) is a relatively long level that has a number of the characteristic hallway formations of the original. In contrast to the original, it ends with a hallway rather than a courtyard or chamber. These differences result in a combination of relatively higher inverse match ($1 - match$) value and relatively lower *dist* and *misalign* values than other levels.
- Level G ($I = 0.95$) also comprises a variety of basic and compound shapes and NPCs. It still ends with an NPC to fight in a chamber. However, it is a long level compared to the original level, and has many more courtyards.
- Level H ($I = 0.95$) also comprises a variety of basic and compound shapes and NPCs. It still ends with an NPC to fight in a chamber. It is of a similar length to the original level but has a number of unique zig-zag hallway segments.
- Level J ($I = 0.95$) is similar length to original and ends with an NPC to fight. Unlike the original it has two objects to collect.
- Level K ($I = 0.97$) comprises a variety of basic and compound shapes interspersed with NPCs. It is of a similar length to the original but has one courtyard with an object to collect, one at end with NPC to fight.
- In the second bracket fall designs E:
- Level E ($I = 0.89$) is similar in length to the original and ends with an object to collect, but there seems nothing else to particularly distinguish it from the original. As a result it has a slightly lower novelty value and falls out of the highest creativity bracket discussed here.
- In the third bracket fall designs B and L.
- Levels B ($I = 0.70$) is similar in length to the original and ends with an NPC to fight. However, there seems nothing else to particularly distinguish it from the original, resulting in a low novelty value and lower interest value than other levels.
- Level L ($I = 0.70$) is relatively long compared to the original and captures a number of the corridor formations found in the original level. It thus has a lower novelty value and lower creativ-

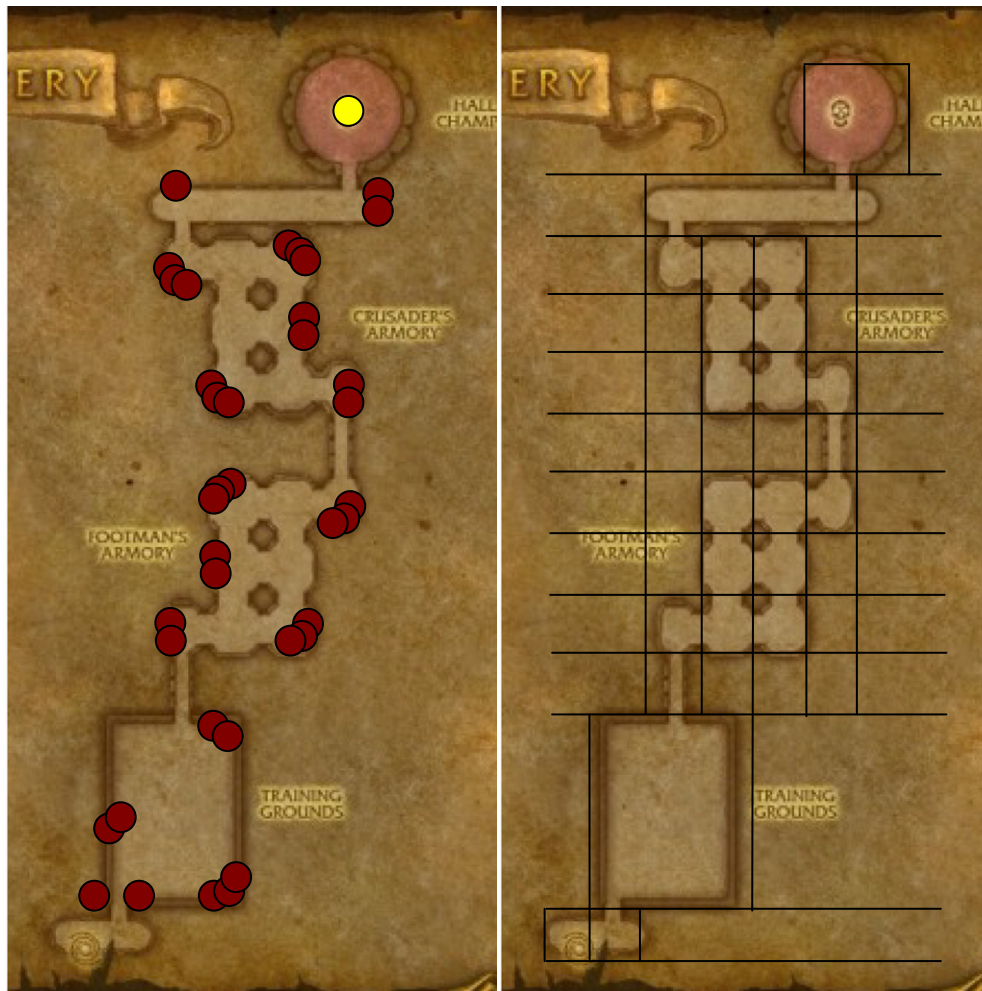


Fig. 6. Plan view of the Scarlet Monastery Armory from *World of Warcraft*. Image from http://www.wowwiki.com/Scarlet_Monastery. (a) Red circles show the locations of common NPCs. The yellow circle indicates a rare NPC (boss). (b) A rough decomposition of the armory using a 2D grid. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

ity value. This level also has no NPC to fight at the end, resulting in a possible anticlimax. It thus seems correctly classified at a lower interest.

- These computational interest values provide a guide as to which generated levels may be most suitable as new instances in the game. The human game designer can choose a creativity threshold above which game instances are of sufficient quality to incorporate in a live game. In each case, it is assumed that the elements of the original design can now be overlaid on the new plans generated by the system to produce a complete, new instance.

5. Conclusion and future work

This paper has presented a framework for PCG systems that use computational models of creativity as a part of the generative process. We have demonstrated an example of such a system that combines the generative shape grammar formalism with a model of creativity based on the Wundt curve to select new designs that are similar-yet-different to existing human designs. The approach aims to capture the usefulness and value of existing designs while introducing novel variations. We introduced an extensible distance metric based on a game content taxonomy as a means to compute novelty.

Results from our experiments and case study generating combat instances for a MMORPG suggest that:

A novelty function can be defined in terms of a variety of difference metrics to distinguish between designs generated by an artificial system.

A computational model of creativity based on novelty and interest may be able to provide the designer with feedback on which generated designs represent more creative new game instances.

The *Shape Grammar Designer* system demonstrated in this paper is a first step towards a PCG system that can automate the process of analysing and decomposing an existing design, extracting design knowledge and creating new designs. Specifically, we focus on automating the process of creating new designs in this paper. A number of areas for future work are revealed by this study, and are discussed in the following sections.

5.1. Automation of the creative process

Fig. 1 illustrated our high level framework for using computational models of creativity in PCG. To permit the evaluation of creativity to occur in a social context, our framework assumes the existence of at least one initial human design as a starting point for the creative process. This design is then decomposed (Step 1)

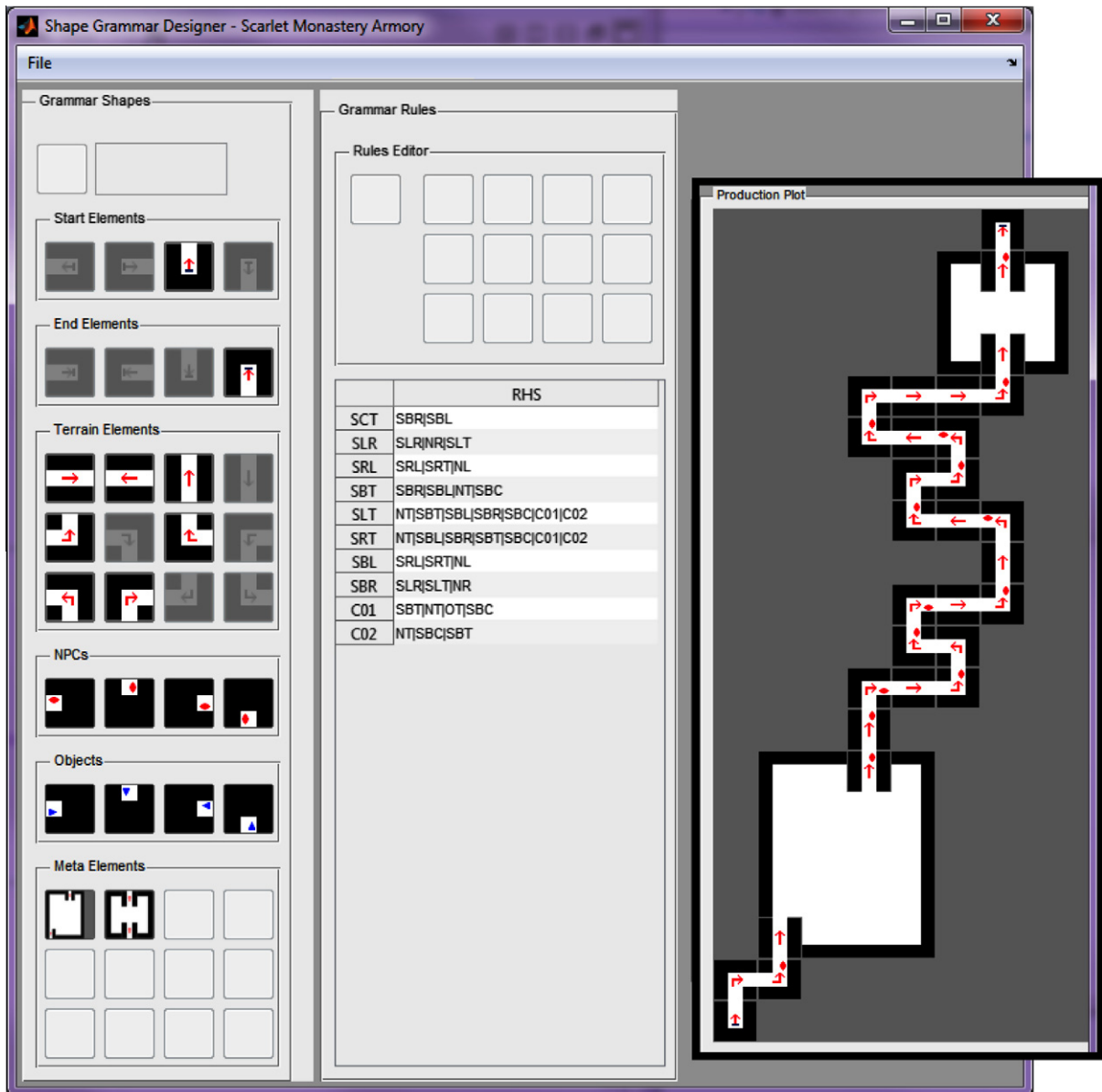


Fig. 7. Decomposition of the *Scarlet Monastery Armory* using a shape grammar and (inset) a demonstration that this grammar can produce a design very similar to the original (low creativity).

into primitive components that become the design variables input to the creative process (Step II). The creative process recombines those components to create new designs. This paper presents a tool that aids the human designer in Step I and automates the role of the human in Step II. Automation of Step I is thus an area for future work.

An approach to this task is likely to integrate closely with the design tools used to construct game levels. Such a system will need to identify logical groups of game content based on some taxonomy and use these to form basic and compound shapes within the generative system. Teleological models that may be applied to these tasks include function–behavior–structure (FBS) [50] and structure–behavior–function (SBF) [51] formalisms. These teleological models have been used to understand a range of complex systems including engineered systems with multiple connected components and causal connections with multiple levels of abstraction. They are thus a potential means by which we can understand, model and decompose designed systems such as computer games.

Automating Step I to produce a grammar (or other) representation of a specific design will also provide a basis for artificial systems that can go beyond a specific design language. Once a design language has been produced for a given individual or set of initial designs, transformations of the language using rule addition, rule deletion or rule change are possible. This means that a shape grammar can be transformed into a new shape grammar, potentially increase the creative potential of Step II by permitting evolution of new design languages.

5.2. The role of experience in creativity

In this paper, the model of the experiences of the artificial system was limited to a single design generated by a human designer. However a number of other models are possible. Where there are multiple samples of human designs in a particular style, for example, a cumulative representation of these designs may be used to represent the experiences of the agents. One approach to this that has shown promise at the intersection of cognitive sci-

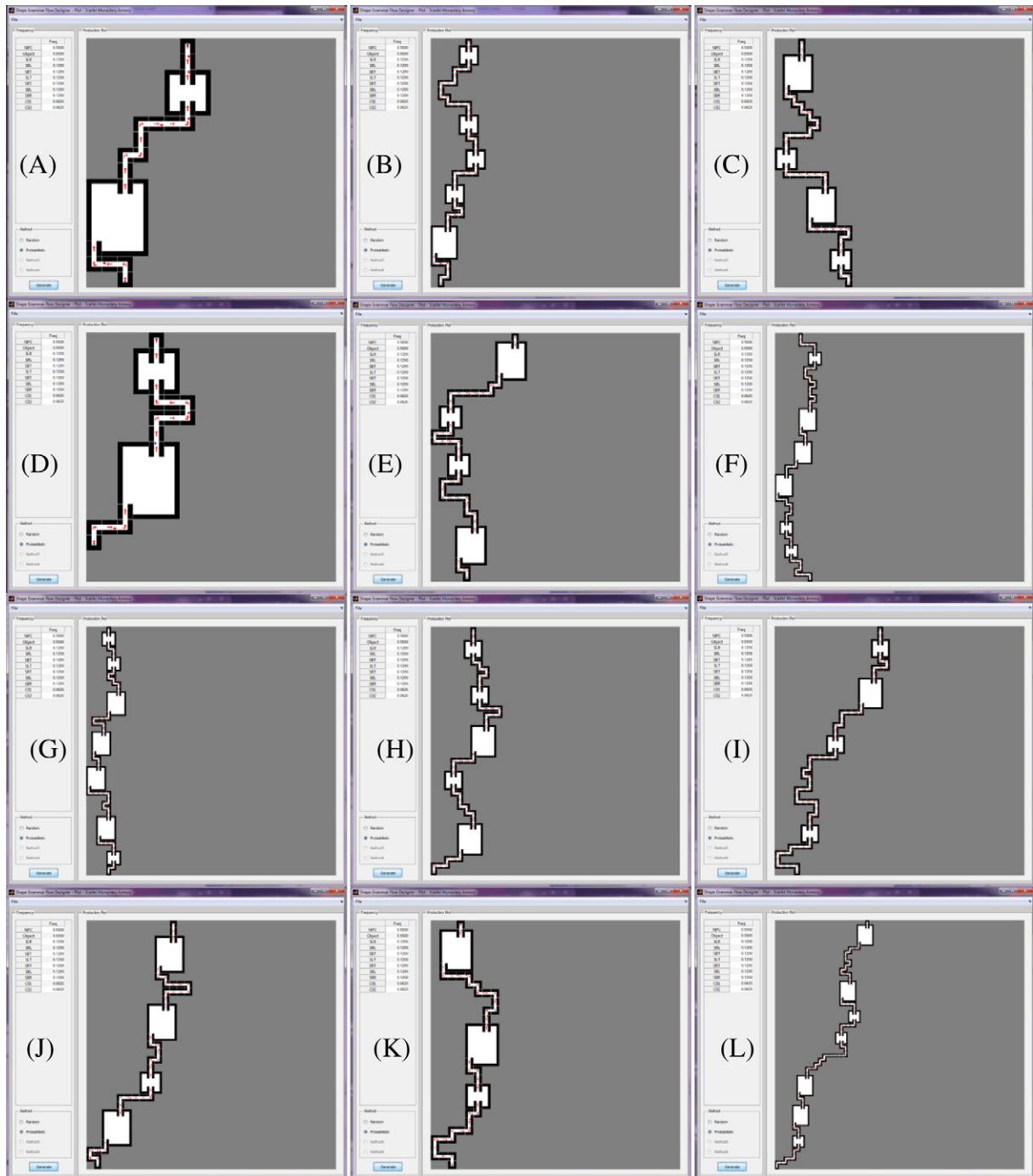


Fig. 8. Twelve system generated designs (Design IDs: A–L).

ence and gaming literature [22] is to use an unsupervised learning algorithm such as a self organizing map (SOM). A SOM can learn a cumulative representation of parameterized designs and this cumulative representation can be used in place of the reference design D_r .

Another approach to representing experiences that could be used independently or in conjunction with the approach described above is for the agent to build a cumulative representation of its own creative designs. This cumulative representation can again be used in place of the reference design D_r so that the creativity of newly generated designs is evaluated based on the sequence

of designs previously generated by the agents. This means that the agent's perception of its own creativity will change based on the specific designs it has already generated. This captures the concept of changing personal creativity (P-creativity) discussed by Boden [3].

A third approach to representing experiences could further capture Boden's [3] concept of historical creativity (H-creativity). Unlike P-creativity, H-creativity depends on the experiences of other individuals in a creative society. In the context of a PCG application these might be the individual or cumulative experiences of both humans and other artificial design agents.

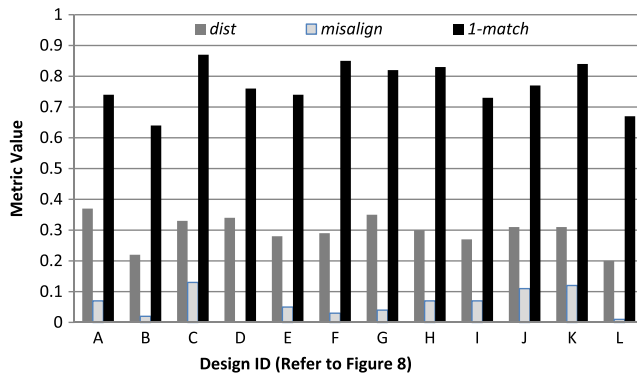


Fig. 9. Difference metric values for the twelve system-generated designs A-L shown in Fig. 8.

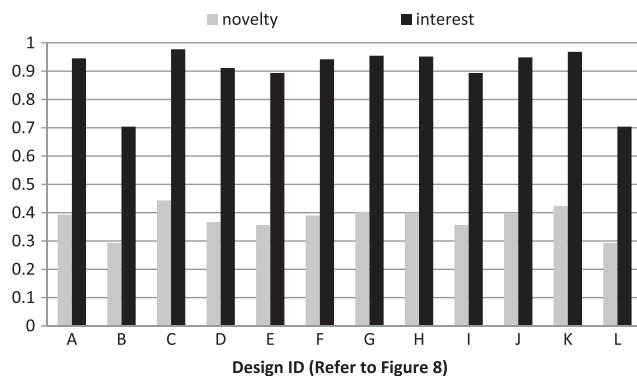


Fig. 10. Novelty and interest values for the twelve system-generated designs A-L shown in Fig. 8.

5.3. Procedural content generation, computational creativity and complex game content taxonomies

This paper focused on PCG within a fairly narrow scope, specifically instance spaces for combat quests in MMORPGs. The self-contained nature of instances, the ability to separate quest and space, as well as the need for continuous development of MMORPGs to provide players with new avenues for character development, makes them an ideal candidate for PCG. Other work has focused on PCG of other aspects of game content and in other genres [1,2,41]. In future it is thus conceivable that PCG systems incorporating computational models of creativity should also be possible across a range of game content types and genres. A range of new research questions arise in this case, including how to model and evaluate creativity across varying game content that has potentially been generated by a range of different generative algorithms.

References

- [1] J. Togelius, G. Yannakakis, K. Stanley, C. Browne, Search based procedural content generation: a taxonomy and survey, *IEEE Transactions on Computational Intelligence and AI in Games* 3 (2011) 172–186.
- [2] M. Hendrikx, S. Meijer, J. Van Der Velden, A. Iosup, Procedural content generation for games: a survey, *ACM Transactions on Multimedia Computing Communications and Applications* (2011).
- [3] M. Boden, *The Creative Mind: Myths and Mechanisms*, 2nd ed., Routledge, 2003.
- [4] J.S. Gero, Creativity, emergence and evolution in design, in: *Second International Roundtable Conference on Computational Models of Creative Design*, Heron Island, 1992, pp. 1–28.
- [5] Esri, Esri CityEngine 2011, <<http://www.esri.com/software/cityengine/index.html>>, 2012 (accessed 12/3/2012).
- [6] P. Goswell, J. Jo, Real-time 3D city generation using shape grammars with LOD variations, *World Academy of Science, Engineering and Technology* 61 (2012) 355–361.
- [7] M.L. Maher, Creative design using a genetic algorithm, *Computing in Civil Engineering* (1994) 163–2021.
- [8] L. Alem, M.L. Maher, A model of creative design using a genetic metaphor, in: T. Dartnall (Ed.), *Artificial Intelligence and Creativity: An Interdisciplinary Approach*, Kluwer, 1994, pp. 281–291.
- [9] A. Goel, Design, analogy and creativity, *IEEE Intelligent Systems* 12 (1997) 62–70.
- [10] M.L. Maher, B. Balachandran, Flexible retrieval strategies for case-based design, in: J.S. Gero (Ed.), *Artificial Intelligence in Design*, Kluwer Academic Press, 1994, pp. 163–180.
- [11] R. Saunders, *Curious Design Agents and Artificial Creativity*, Faculty of Architecture, University of Sydney, Ph.D. Thesis, 2001.
- [12] J.S. Gero, Understanding situated design computing: Newton, Mach, Einstein and quantum mechanics, in: I.F. Smith (Ed.), *Intelligent Computing in Engineering and Architecture*, Springer, Berlin, 2006, pp. 285–297.
- [13] J.S. Gero, U. Kannengiesser, An ontological model of emergent design in software engineering, Presented at the ICED07, Ecole Centrale de Paris, 2007.
- [14] W. Byrne, T. Schnier, R. Hendley, Computational intelligence and case-based creativity in design, Presented at The Fifth International Joint Workshop on Computational Creativity, Madrid, 2008.
- [15] M.L. Maher, B. Balachandran, D. Zhang, *Case-Based Reasoning in Design*, Lawrence Erlbaum Associates, 1995.
- [16] A. Liapis, G. Yannakakis, Neuroevolutionary constrained optimization for content creation, Presented at The 2011 Conference on Computational Intelligence and Games, 2011.
- [17] J. Secretan, N. Beato, D. D'Ambrosio, A. Rodriguez, A. Campbell, K. Stanley, Picbreeder: evolving pictures collaboratively online, Presented at the CHI 2008, Florence, Italy, 2008.
- [18] J. Schmidhuber, Developmental robotics, optimal artificial curiosity, creativity, music and the fine arts, *Connection Science* 18 (2006) 173–187.
- [19] P.-Y. Oudeyer, F. Kaplan, Intelligent adaptive curiosity: a source of self-development, Presented at The Fourth International Workshop on Epigenetic Robotics, 2004.
- [20] C. Browne, *Automatic Generation and Evaluation of Recombination Games*, Faculty of Information Technology, Queensland University of Technology, Ph.D. Thesis, 2008.
- [21] G. Yannakakis, J. Hallam, A generic approach for obtaining higher entertainment in predator/prey computer games, *Journal of Game Development* 1 (2005) 23–50.
- [22] K. Merrick, M.L. Maher, *Motivated Reinforcement Learning: Curious Characters for Multiuser Games*, Springer, Berlin, 2009.
- [23] J. Heckhausen, H. Heckhausen, *Motivation and Action*, Cambridge University Press, New York, 2008.
- [24] W. Wundt, *Principles of Physiological Psychology*, Macmillan, New York, 1910.
- [25] R. Saunders, *Curious Design Agents and Artificial Creativity*, Ph.D., Faculty of Architecture, University of Sydney, Sydney, 2001.
- [26] S. Marsland, U. Nehmzow, J. Shapiro, A real-time novelty detector for a mobile robot, in: *EUREL European Advanced Robotics Systems Masterclass and Conference*, University of Salford, Manchester, UK, 2000.
- [27] T.W. Knight, Shape grammars in education and practice: history and prospects, *International Journal of Design Computing* 2 (2000) (online journal).
- [28] G. Stiny, Introduction to shape grammars, *Environment and Planning B* 7 (1980) 343–351.
- [29] T.W. Knight, *Transformations in Design: A Formal Approach to Stylistic Change and Innovation in the Visual Arts*, Cambridge University Press, England, 1994.
- [30] G. Stiny, W.J. Mitchell, The Palladian grammar, *Environment and Planning B* 5 (1978) 5–18.
- [31] G. Stiny, W.J. Mitchell, The grammar of paradise: on the generation of Mughul Gardens, *Environment and Planning B* 7 (1980) 209–226.
- [32] H. Koning, J. Eizenberg, The language of the prairie: Frank Lloyd Wright's prairie houses, *Environment and Planning B* 8 (1981) 295–323.
- [33] T.W. Knight, Applications in architectural design, education and practice, Presented at the NSF/MIT Workshop on Shape Computation, 1999.
- [34] T.W. Knight, Designing a shape grammar, Presented at the *Artificial Intelligence in Design*, Netherlands, 1998.
- [35] G. Yannakakis, J. Togelius, Experience-drive procedural content generation, *IEEE Transactions on Affective Computing* 2 (2011) 147–161.
- [36] R. Lopes, R. Bidarra, Adaptivity challenges in games and simulations: a survey, *IEEE Transactions on Computational Intelligence and AI in Games* 3 (2011) 85–99.
- [37] D. Ramirez-Cano, S. Colton, R. Baumgarten, Player classification using a meta-clustering approach, Presented at The Third Annual International Conference on Computer Games, Media and Allied Technology, Singapore, 2010.
- [38] J. Chen, Flow in games (and everything else), *Communications of the ACM* 50 (2007) 31–34.
- [39] R. Smelik, T. Tutenel, K. de Kraker, R. Bidarra, A declarative approach to procedural modeling of virtual worlds, *Computers & Graphics* 35 (2011) 352–363.
- [40] G. Smith, J. Whitehead, M. Mateas, Tanagra: a mixed-initiative level design tool, Presented at The Fifth International Conference on the Foundations of Digital Games, New York, 2010.

- [41] J. Dormans, Adventures in level design: generating missions and spaces for action adventure games, Presented at The 2010 Workshop on Procedural Content Generation in Games, 2010.
- [42] M. Duggan, The Official Guide to 3D Game Studio, Thomson Course Technology, Boston, MA, 2007.
- [43] T. Meigs, Ultimate Game Design: Building Game Worlds, McGraw-Hill/Osborne, Emeryville, CA, 2003.
- [44] G. Oh, J.Y. Kim, Effective quest design in MMORPG environment, Presented at the Game Developers Conference 2005, 2005.
- [45] A. Sullivan, Gender-inclusive quest design in massively multiplayer online role-playing games, Presented at The Fourth International Conference on Foundations of Digital Games, 2009.
- [46] R. Bellman, Dynamic Programming, Princeton University Press, Princeton, NJ, 1957.
- [47] H. Sakoe, S. Chiba, Dynamic programming algorithm optimisation for spoken word recognition, IEEE Transactions on Acoustics, Speech and Signal Processing 26 (1978) 43–49.
- [48] M. Barlow, M. Wagner, Measuring the dynamic encoding of speaker identify and dialect in prosodic parameters, Presented at the ICSLP, 1998.
- [49] Blizzard, World of Warcraft, Released November 23, 2004. <www.worldofwarcraft.com>, 2004 (accessed March, 2012).
- [50] J.S. Gero, Design prototypes: a knowledge representation schema for design, AI Magazine 11 (1990) 26–36.
- [51] A. Goel, S. Raugaber, S. Vattam, Structure, behavior and function of complex systems: the structure, behaviour and function modeling language, Artificial Intelligence for Engineering Design, Analysis and Manufacturing 23 (2009).