3D Game Model and Texture Generation using Interactive Genetic Algorithm

DuMim Yoon*
Dept. of Computer Engineering, Sejong University

Kyung-Joong Kim†
Dept. of Computer Engineering, Sejong University

Abstract

Recently, the production of big games usually requires lots of money but it is very difficult to gain attention from players. Although gamers expectations are very high, game companies resource is limited to maximize the quality of the games. Mod (game contents editing) can be one of the solutions to this problem. It makes gamers satisfy themselves from creating and sharing their Mod. It increases gamers playing time and sales. But there are only few users who have knowledge and special ability to create Mod. So most gamers just use Mod made by someone else. In this paper, we propose a method to generate Mod for 3D game objects and textures. Our method enables the construction of a 3-dimension object using the grown building footprints by L-system, and it also provides a web-based interactive genetic algorithm interface to users for changing shape. Furthermore, it also provides a function to evolve various texture images from an original texture file. We demonstrated the possibility of our systems using TORCS (The Open Racing Car Simulator). These results reveal that 3D objects and textures can be created from our method by amateur users.

CR Categories: I.2.1 [Artificial Intelligence]: Applications and Expert Systems—Games

Keywords: 3D model, game contents generation, interactive generic algorithm

1 Introduction

Video game industry has successfully expanded their territory into personal computer, video game consoles, and mobile devices. It is common that games include fantastic graphics, multi-modal natural interface, and social network play. As a result, the expectation from players is usually very high but development cost for the game increases rapidly. One of the most expensive parts in the game development is contents creation by designers. To reduce the cost, recently, there have been works on procedural game contents generation which uses algorithmic procedure to create game contents [Togelius et al. 2011]. In the line of research, it is important to personalize game contents in the automatic generation. Recently, game users have great interest to edit games and company starts to supports the game modification [Bostan 2010]. It opens door to the gamers for the creation of their own personal games by changing shapes, textures and rules of the games. For example, game developers create only parts of planned contents and admit the gamers who have dissatisfied on the original contents to create extra contents by themselves. This approach, called game mod, usually requires high-level skills and semi-expert knowledge [Bostan 2010]. Especially, to create 3 dimensional game contents has been a high barrier for amateurs. Therefore, in this paper, we propose 3D game contents generation based on interactive genetic algorithm (IGA) which creates novel contents based on users feedback. It evolves the 3D shapes of buildings and textures for the structure. The textures are evolved by our photo retouching system [Yoon and Kim 2012]. It allows amateurs to make custom 3D game contents for editable games.

*e-mail:krad@hanmir.com
†e-mail:kimkj@sejong.ac.kr,corresponding author

WASA 2012, Singapore, November 26 – 27, 2012.
© 2012 ACM 978-1-4503-1835-8/12/0011 $15.00

2 Related Works

In this section, we introduce game content generation works based on evolutionary computation. The purpose of the game contents generation using evolution is to find novel and personalized contents automatically. Usually, they adopt the IGA because the evaluation of the solution is subjective and human should be involved in the search process. P. L. Lanzi et al. used the interactive genetic algorithm to generate novel racing tracks for a simulated car racing game (http://trackgen.pierlucalanzi.net/) [Loiacono et al. 2011]. This method generates novel game tracks based on anonymous users evaluation (positive or negative) counts via web. It is possible to download the evolved tracks and applied them into the game. K. Stanely et al. evolved a bullet pattern using artificial neural networks in their shooting games (Galactic Arms Race) (http://gar.eecs.ucf.edu/) [Hastings and Stanley 2010]. Their results show that it is possible to create interesting and successful bullet patterns automatically. It can save the effort for game developers to design effective and interesting attack patterns.

3 3D Building Model Generation

Although background buildings are minor content than characters, it definitely needs in urban background games. It is important to increase the diversity of buildings in the background to improve user experience. In this section, we introduce the automatic building structure evolution.

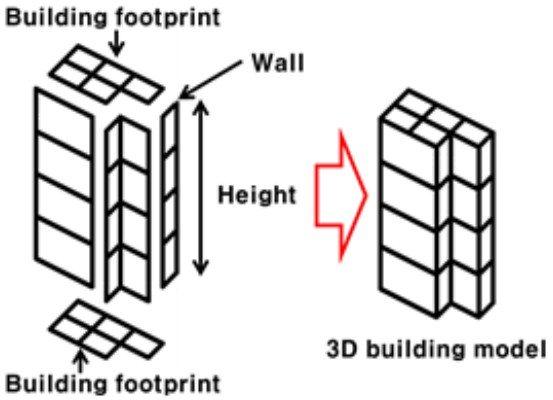
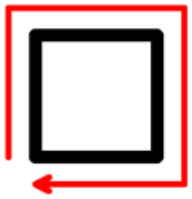


Figure 1: *Structure of 3D building model.*

3.1 Making a building footprint using L-system

In this paper, we generate arbitrary building shape by growing a footprint of building (Figure 1). Although there are some works

on automatic building design system [Müller et al. 2006][Wonka et al. 2003], we focus on using IGA for game contents. To represent the footprint, we adopt L-system [Prusinkiewicz and Lindenmayer 1990] that is powerful to generate complex shapes using simple grammars. It generates a randomized result based on a seed value using a reproducible random number generator. In this work, we use the random number generator by George Marsaglia [Marsaglia 2003]. Initially, a square building footprint is generated and it grows to other shapes by the L-system grammatical rules. Although the starting shape can be angular, circular and polygonal shapes, but we test only the square type. The grammatical rule for the L-system is written with R, L and F. These three characters, R, L and F, mean clock-wise rotation 90, -90 and forward. The R and L just change the direction of drawing. The F is actually drawing a single unit line to the current direction.



(FRFRFRFR)

Production Rule
"F" → "LFRFRFL"

An Example (stochastic L-system)
"FRFRFRFR" → "FRFRFRLFRFRFLR"

Figure 2: *Initial building footprint (top), grammar rule (middle) and an example (bottom)*

The starting point of our stochastic L-system is defined as a string "FRFRFRFR" which draws a square (Figure 2). The production rule is F→LFRFRFL defined manually. In the production, for each F in the string, we get a random value ranged from 0 to 1 and if the value is smaller than a pre-defined probability threshold (randomly chosen), the rule is applied. In the example, only one F is replaced with "LFRFRFL."

3.2 Modifying Results

From the grammatical production, we can get a building footprint represented as a string of the alphabets. It shows that "FRRF" or "FLLF" result in a single line instead of polygons. The strings are removed during the production. Besides the two strings, there are several redundant information in the string and can be shortened (Table 1).

Table 1: *Modification on string*

Original string	Replacement
"FLLF"	"LL"
"FRRF"	"RR"
"RRRR"	""
"LLL"	"R"
"RRR"	"L"
"LL"	"RR"
"RL"	""
"LR"	""
"FRFRFRF"	"L"
"FLFLFLF"	"R"

The modification is deterministic and all the substrings matched to the pattern are replaced with the alternatives.

3.3 Web-based Evolution

Because the building designs are for the games, the evaluation on them is subjective. The interactive evolutionary computation is a useful tool to search for solutions based on subjective evaluation. Usually, it maintains the small size of populations and users are involved in the evaluation. Based on the scores from users, the genetic algorithm creates a new generation from the parents. Because of users fatigue, it is important to get reasonable solution with small number of evaluations. Traditionally, the Interactive Genetic Algorithm is running in a personal computer. However, the new web-based environments allow user evaluates or evolves solution anywhere and anytime. If users can access the internet from their smart phone, it is possible to run the evolution with the device. Also, if the evaluation is open to the public, it can search for solution with collective intelligence. There are several examples exploiting votes from anonymous users. For example, users can evolve 3D objects through webpage and finally they can be printed using 3D printer (http://endlessforms.com). Our interface is running on web and anyone can access to the program to evolve building footprints (http://cilab.sejong.ac.kr/EC)[1] . In this work, the chromosome is consisted of values for a seed for RNG (Random Number Generator), the iteration number in the L-system production (0∼10), and the height of the building (0∼15). The population size is 12. After the initialization of the population, the interface shows 3D buildings generated with the default texture. By clicking the building, user can give scores and get a corresponding 3D file exportable to 3D simulated car racing games (for example, TORCS). The evolution continues by creating a new population based on user evaluation and genetic operations. If user selects a building, the score is one and otherwise zero. It means that the chromosome scored as zero has no chance to survive in the next generation. Although the individual is not selected, there is a little chance that the parts of the individuals chromosome are useful to create novel solutions. The following equation is used to adjust the original fitness value (in this paper, K=3).

$$f_i = (S_i - S_w) + \frac{S_b - S_w}{K - 1}, (K > 1) \quad (1)$$

fi : i-th fitness
Si : i-th score
Sb : the best score
Sw : the worst score
K : Selection Pressure

The chromosome is presented with 6 bytes (4 bytes for the seed, 1 byte for the number of iterations in the production, and 1 byte for the height). It adopts a standard one-point crossover. The mutation operator is applied to the chromosome in the byte level. For each byte of the chromosome is tested whether the mutation is applied or not with small mutation probability (in this paper, 3%). In previous works, researchers attempt to creates 2D contents or a single-colored 3D contents [Clune and Lipson 2011][Cardamone et al. 2011][Secretan et al. 2008]. In this work, the interface is used to evolve 3D buildings. Although the texture of the building is not evolved together with the building structure, we also provide separate systems to evolve the texture of the game contents. The system uses HTML 5 and WEBGL. It allows rendering 3D objects in users web browsers. The Figure 3 (a) shows a set of successful 3D building models and their average score. The interface can export the successful building structure to AC3D format.

[1]Google Chrome, Mozilla Firefox, Safari, and Opera platforms

(a) Evolved contents

(b) A snapshot in the evolution (Green means the selection by a user)

```
AC3Db
MATERIAL "ac3dmat1" rgb 1 1 1 amb 0.2 0.2 0.2
emis 0 0 0 spec 0.5 0.5 0.5 shi 10 trans 0
OBJECT world
kids 20
OBJECT poly
name "poly"
loc 0 0 0
texture "building1.rgb"
numvert 4
0 0 0
0 10 0
...
```

Figure 3: *Some snapshots from the interface and the exported 3D contents for games*

4 Texture Retouching by IGA

Using the web-based interface, users can generate 3D structure of the buildings. The interactive evolution searches for the shape of the buildings footprint and height. However, in the evolution, the texture of the building is not changed. At this moment, the evolution of the 3D model and its texture is not integrated as a single system. In [Yoon and Kim 2012], authors developed an IGA-based system to evolve a set of filters to be applied to the input image. The interface shows twelve new images filtered from the input. It also evolves the number of filters to be applied to the target. It starts from a single filter but the genetic operators add or delete filters from the initial filter sets. Unlike the previous evolution of the 3D model, this evolution assumes that the length of the chromosome is not fixed. The length is proportionate to the number of filters used. For example, a solution can be a sequence of filters (for example, filter 1, filter 5, filter 6 and filter 3). The filters are applied to the original input image and the output image is passed to the next filter. In commercial photo edit tools, they support a lot of filters (smoothing, sharpening, special effects and so on). However, theyre not useful for our filter evolution software because they dont provide interface to automate the sequential filtering. GIMP (GNU Image Manipulation Program) supports a script-based interface to process images and they can be combined with our IGA software. Our system generates a GIMP script file dictating the sequence of filters to be applied to the input images and pass it to the GIMP. The program outputs the results of the sequential filtering. Finally, our program visualizes the number of new images filtered and users give scores for the goodness of the processing. Usually, it is not easy to create novel images from scratch but the retouching system can generate lots of interesting figures because it starts from user-created photos or patterns (Figure 4).

4.1 Script-Fu

GNU Image Manipulation Program (GIMP) is a cross-platform image editor tool. It supports drawing, editing and special effects including various filters. Also, it has an interface to do batch processing called script-fu. Because of the property, the program has been used in several research works [Petcu and Iordan 2006][Bucior and Ventures 2007]. For example, GIMP was used to create artistic image [Valente and Klette 2010], sharpen image [Jaksa and Takagi 2003], and lighten image [Hara et al. 2009]. In this work, the script-fu (script-based interface for the GIMP) is used to combine a set of filters to produce new image from an original one (Figure 5).

4.2 Evolution

Figure 6 shows an interface for the image retouching. Initially, it generates twelve filtered images and waits for users feedback on them. For each image, user can express preference as one of "good" or "bad." For the "good" decision, the fitness value is one. Otherwise, the value is zero. It also supports sliding scale input, and five stars. Filters have their parameters to adjust the result. The chromosome also contains information on the details of parameters for each filter. The number of parameter is different for each filter type. However, it ranges from 3 to 5. It should have the order of filters and their parameters. The chromosome is automatically converted into a script-fu for GIMP. Because the number of filters cannot be determined a priori, our evolutionary system uses a variable-length chromosome. It starts from empty chromosome, but it gradually increases the length from mutation operations.

For each filter, four bytes are assigned to represent parameters and filter type. In total, 40 image filters are used (Table 2). Theyre color balance, copy layer, loop and so on. However, we exclude several filters and complex functions such as a 3x3 mask because errors occur. The first byte is used to represent the type of filter (0∼39). The other three bytes are used to represent parameters of the filter. If the number of filters for a chromosome is N, the size is 4N bytes.

If the number of parameters for the filter is three, each parameter is represented with one byte. However, some filters have four or five parameters. In the case, one or two bytes are divided into smaller

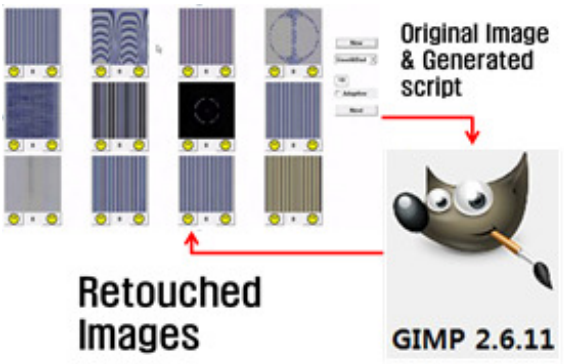


Figure 4: *An overview of the texture retouching system*

```
(define (custom-filterID srcfilename destfilename)
(let* ((image (car (gimp-file-load RUN-
NONINTERACTIVE srcfilename srcfilename)))
(drawable (car (gimp-image-get-active-layer image))))
(if (not (= RGB (car (gimp-image-base-type image))))
(gimp-image-convert-rgb image))
(gimp-message "processingID")
...
(gimp-image-flatten image)
(gimp-file-save RUN-NONINTERACTIVE image (car
( gimp-image-get-active-layer image)) destfilename
destfilename)
(gimp-image-delete image) ))
```

Figure 5: *A template of script-fu for the filter evolution. Each filter set in the population has a unique ID. In the middle of the script-fu code, a set of codes for filtering is added.*

parts to represent more than two parameters (Figure 7). Unlike the 3D model chromosome, the texture evolution uses variable-length chromosomes. In the initialization, each chromosome assigns one filter randomly chosen. Because the mutation operator allows addition and deletion of filters in the chromosome, the length of the filter sets can be changed. It uses a one-point crossover and mutation operators.

4.3 Interface

The communication between the GIMP and the IGA interface is done using file I/O. Each chromosome is converted into a script-fu file and the GIMP outputs one result image file. In the interface,

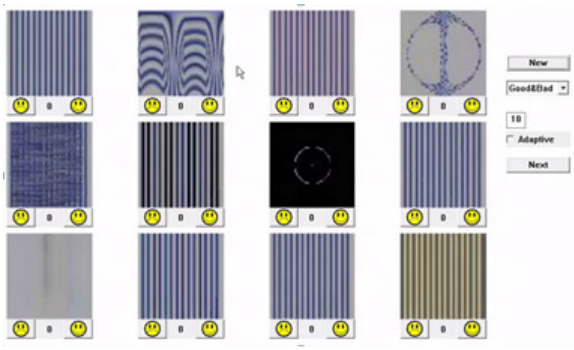


Figure 6: *Image retouching interface*

Table 2: *GIMP image filters*

Blur	gblur,mblur,pixelize
Enhance	antialias, deinterlace, destripe, nlfilter, red-eye-removal, sharpen, unsharp-mask
Distorts	dog, edge, laplace, neon, sobel, dilate, erode, apply-canvas, cartoon, cubism, oilify, photocopy, soft-glow, blinds, emboss, lens-distortion, polar-coords, ripple, shift, vpropagate, video, waves, whirlpitch, wind
Etc	color-balance, levels, invert, flip, copy-layer, loop

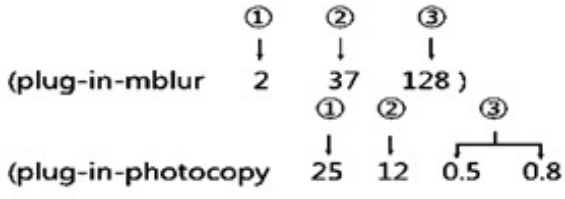


Figure 7: *An example of chromosome representation (For plug-in-photocopy, the last byte is divided into four bits to represent two parameters in a byte.)*

users select their own photos or patterns to create new textures for 3D model. The evaluation by user can be converted to the numeric value. For example, the five star input can be interpreted as 0.0, 0.2, 0.4, 0.6, 0.8, and 1.0. Slide scale represents a real value from 0 to 1. Figure 8 shows an example images created from the IGA evolution. It starts from an original image and a set of filters can transform it to a variety of novel textures.

5 Applying to a Real Game

In this section, we show that the new contents can be used in racing games (TORCS, The open Racing Car Simulator). Because this game uses 3D model format (AC3D) and track toolkit, it is an easily editable game. The web-interface can export the 3D building model in AC3D format. Users just simply download the converted file and insert it into the game folder. The texture evolved was used for the buildings. Because the TORCS only accepts RGB format, we converted the image with GIMP. Figure 9 shows the final racing screen shots with the evolved buildings and textures.

6 Conclusions and Future Works

In this paper, we propose a 3D model and texture generating method that enables amateur to make new contents for games using IGA. In

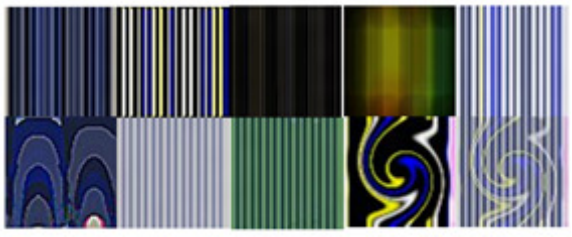


Figure 8: *Some example images from GIMP batch processing*

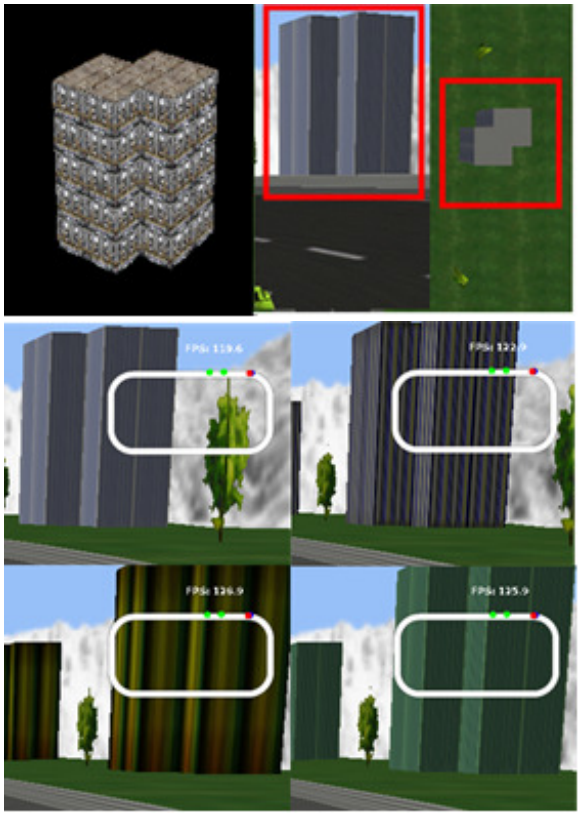


Figure 9: *Running TORCS games with the created contents*

this work, we still have a lot of constraints (for example, fixed initial building footprint size) to generate complex 3D building models. But it shows that the IGA is promising to produce 3D models automatically from the feedback by users. The new contents generation system allows users to create their own game contents and share them with others. We plan to generate more complex 3D buildings and extend our system to other types of game objects (such as cars, characters, and so on). Also, we need to integrate the structure and texture evolution into a single system.

7 Acknowledgements

This research was supported by Basic Science Research Program and the Original Technology Research Program for Brain Science through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0012876) (2010-0018948)

References

BOSTAN, B. 2010. Explorations in player motivations: virtual agents. In *Proceedings of the 9th international conference on Entertainment computing*, Springer-Verlag, Berlin, Heidelberg, ICEC'10, 262–269.

BUCIOR, B., AND VENTURES, S., 2007. Using script-fu in the gnu image manipulation program to automate "smart" sharpening.

CARDAMONE, L., LOIACONO, D., AND LANZI, P. L. 2011. Interactive evolution for the procedural generation of tracks in a high-end racing game. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, ACM, New York, NY, USA, GECCO '11, 395–402.

CLUNE, J., AND LIPSON, H. 2011. Evolving 3d objects with a generative encoding inspired by developmental biology. *SIGEVOlution 5*, 4 (Nov.), 2–12.

HARA, K., MAEDA, A., INAGAKI, H., KOBAYASHI, M., AND ABE, M. 2009. Preferred color reproduction based on personal histogram transformation. *Consumer Electronics, IEEE Transactions on 55*, 2 (may), 855 –863.

HASTINGS, E. J., AND STANLEY, K. O. 2010. Interactive genetic engineering of evolved video game content. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, ACM, New York, NY, USA, PCGames '10, 8:1–8:4.

JAKSA, R. NAKANO, S., AND TAKAGI, H. 2003. Image filter design with interactive evolutionary computation. In *Proceedings of the IEEE International Conference on Computational Cybernetics*, 1–6.

LOIACONO, D., CARDAMONE, L., AND LANZI, P. 2011. Automatic track generation for high-end racing games using evolutionary computation. *Computational Intelligence and AI in Games, IEEE Transactions on 3*, 3 (sept.), 245 –259.

MARSAGLIA, G. 2003. Random number generators. *Journal of Modern Applied Statistical Methods 2*, 1 (May), 2–13.

MÜLLER, P., WONKA, P., HAEGLER, S., ULMER, A., AND VAN GOOL, L. 2006. Procedural modeling of buildings. *ACM Trans. Graph. 25*, 3 (July), 614–623.

PETCU, D., AND IORDAN, V. 2006. Grid service based on gimp for processing remote sensing images. In *Proceedings of the Eighth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, IEEE Computer Society, Washington, DC, USA, SYNASC '06, 251–258.

PRUSINKIEWICZ, P., AND LINDENMAYER, A. 1990. *The algorithmic beauty of plants*. Springer-Verlag New York, Inc., New York, NY, USA.

SECRETAN, J., BEATO, N., D AMBROSIO, D. B., RODRIGUEZ, A., CAMPBELL, A., AND STANLEY, K. O. 2008. Picbreeder: evolving pictures collaboratively online. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, ACM, New York, NY, USA, CHI '08, 1759–1768.

TOGELIUS, J., KASTBJERG, E., SCHEDL, D., AND YANNAKAKIS, G. N. 2011. What is procedural content generation?: Mario on the borderline. In *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*, ACM, New York, NY, USA, PCGames '11, 3:1–3:6.

VALENTE, C., AND KLETTE, R. 2010. Artistic emulation - filter blending for painterly rendering. In *Proceedings of the 2010 Fourth Pacific-Rim Symposium on Image and Video Technology*, IEEE Computer Society, Washington, DC, USA, PSIVT '10, 462–467.

WONKA, P., WIMMER, M., SILLION, F., AND RIBARSKY, W. 2003. Instant architecture. *ACM Trans. Graph. 22*, 3 (July), 669–677.

YOON, D.-M., AND KIM, K.-J. 2012. Comparison of scoring methods for interactive evolutionary computation based image retouching system. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation con-*

ference companion, ACM, New York, NY, USA, GECCO Companion '12, 617–618.