Technologies of reconstruction and procedural generation of three-dimensional content

Alexander Mezhenin¹, Vera Izvozchikova², Vladimir Shardaakov² and Alexey Korotkikh¹

¹Saint Petersburg National Research University of Information Technologies, Mechanics and Optics (ITMO University), Kronverksky ave. 49, St. Petersburg, 197101, Russia
²Orenburg State University, ave. Pobedy 13, Orenburg, 460018, Russia

E-mail: mejenin@mail.ru

Abstract. The research is aimed at finding and developing approaches to procedural content generation that meet the conditions of universality and optimality. The use of modular objects is considered as one of the possible solutions and a model for their construction is proposed. This model allows you to create single-tier modular objects, taking into account the possibility of combining modules according to specified rules, which ensures the correctness of their construction. For the proposed model, an algorithm for generating modular objects based on a given contour was developed and implemented. This algorithm provides the ability to generate objects in real time, which allows you to quickly make changes and thereby optimize the process of generating content and make it more flexible.

1. Introduction

To date, the use of procedural generation in the creation of three-dimensional objects has serious limitations presented in the form of insufficient technical capacity of personal computers and effective algorithms to reduce the load on the PC.

The purpose of using procedural generation is to create content without human intervention. The content to be created must meet certain conditions and thus solve the corresponding problems, and in practice the following properties are most often considered:

- Speed: depending on the task requirements vary from milliseconds to months.
- Reliability: ensure that the specified criteria are met.
- Controllability: ability to control generated content based on the situation.
- Diversity: create content that is as different as possible on different launches.
- Creativity and verisimilitude: generating content that looks like it was created by a person, not a generator.

Generation takes place in real time—changes immediately rebuild the created world. This model allows you to determine the scale yourself, adjusting the complexity of scenes and even worlds.

2. Related work

Procedural content generation has long been used to automate production processes in areas related to computer graphics. Procedural content generation refers to automatic and semi-automatic creation, as
well as dynamic change of various types of content [1]. Scientific research of various approaches to
procedural generation is often aimed at its application in engineering fields, but currently such
research is relevant in the development of media products, such as movies, virtual reconstructions,
simulators, computer games and more [1, 2, 3]. Procedural generation allows not only reducing the
resources required to create a large amount of content, but also to make the development process more
flexible and optimal [4, 5, 6]:

Given the large number of task types that procedural generation can be used to solve, there are
many different tools available. For example, Esri CityEngine or CityGenerator (a plugin for UE4),
engaged in the generation of cities. There are also programs of a more General nature that provide the
user with ample opportunities to create procedural content of any type, for example, Houdini or
Substance Designer.

Despite the large number of existing solutions, some of them are often aimed at a narrow range of
tasks (for example, building cities), or Vice versa, gives an extensive set of simple functions that
require users to spend additional resources to create a suitable system [7, 8]. Thus, the actual task
facing between the above-mentioned, that is, the development of an approach to the procedural
generation of content that provides sufficient versatility and requires fewer resources in comparison
with existing analogues. One solution that potentially satisfies these requirements may be the use of
so-called modular objects, which are very often used in practice by computer graphics artists [9].

A modular object is an object consisting of pre-prepared elements (modules) and combined with
each other in a certain way. A modular object can be, for example, a building made up of different
types of walls, Windows, doors, or a bridge consisting of different straight and angular sections.
Creating such objects manually by adding each new element in sequence gives the artist full control
over the final result, which can be important when creating significant objects (objects of the first
plan), but often created objects can be secondary or fairly simple in structure [10]. In this case,
automating the creation of a modular object can significantly optimize the entire process of content
production.

3. Mathematical model

As part of the study, a model was developed that allows to uniquely representing modular objects
consisting of a finite number of elements. The proposed model consists of two main structures:

- Modular object O, which is a sequence of modules \( M = (T, \alpha) \), where \( T \) is the type of module,
and \( \alpha \) is the angle at which the current module is combined with the previous one, and when
combining modules, the start point of the module \( M_{i+1} \) corresponds to the end point of the
module \( M_i \).

- A set of modules representing a set of different types of modules \( T = (I, (x_b, y_b), (x_e, y_e)) \),
where \( I \) is the identifier of the module type, \( (x_b, y_b) \) is the starting point of this type of module
\( (x_e, y_e) \) is the end point module, and the matrix of relations \( R \) of the form:

\[
R = \begin{bmatrix}
C_{11} & C_{12} & \ldots & C_{1n} \\
C_{21} & C_{22} & \ldots & C_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
C_{n1} & C_{n2} & \ldots & C_{nn}
\end{bmatrix},
\]

where \( n \) is the number of modules in the set, \( C_{ij} \) is the set of angles at which a module of type \( j \)
can be combined with a module of type \( i \).

It is important to note that this model is suitable only for single-tier objects. An example of
modules and a modular object is shown in figure 1.
One of the most effective means of controlling and changing the appearance of the generated object is the contour. The contour is a self-intersecting polyline. Thus, one of the tasks to be solved was to develop an algorithm that generates an object along the contour and works in real time.

The proposed solution is based on a greedy algorithm, in which each subsequent module is selected by comparing all available options by the coefficient \( k \). To calculate \( k \), it is necessary to know which of the segments of the contour is relevant for the module being tested (\( B_{c-1} \)). The actual segment is the segment whose number in the loop is greater than or equal to the number of the segment that is relevant to the previous module (\( B_{c-1} \)), and the distance from the end point of the module being tested is minimal (\( d \)). Thus, the coefficient \( k \) is calculated by the following formula:

\[
    k = \begin{cases} 
        d + l^c_e & B_c = B_{c-1}, \\
        d + \sum_{i=B_{c-1}+1}^{B_c} l^b_i & B_c > B_{c-1}, 
    \end{cases}
\]

where \( l^c_e \) - distance from the end point check the module to the point of end of current segment, \( l^b_i \) - distance from the end point check the module to the point of beginning of the i-th segment. The best is the module where the value of \( k \) is minimal.

Given the possibility that the set of modules used may not be sufficient to construct a given path, an algorithm termination condition must be added to prevent an infinite loop. In this case, the algorithm terminates if \( B_c = B_{c-1} \) and \( l^c_e \geq l^b_i \).

It is important to note that this algorithm does not guarantee the most optimal result, which, for example, can provide some algorithms for finding the shortest path, however, unlike them, the complexity of the algorithm is \( O(mn) \), where \( m \) is the number of modules of the generated object, \( n \) is the number of different types of modules taking into account possible angles, which allows it to work in real time. However, a significant disadvantage of this algorithm is the impossibility of guaranteed generation of closed objects.

4. Experimental part
Unreal Engine 4 and Microsoft Visual Studio were used to implement and demonstrate the algorithm. A brief description of the algorithm of the module is presented below:

- Define path and set of modules;
- Check the specified path for no self-intersections;
- for each module available to build:
  - Find the current segment \( B_c \);
  - The calculation of the coefficient \( k \);
Select the module with the lowest value \( k \);
add the selected module to the modular object \( O \);
Repeat steps 3-5 until the following conditions are met:

- For the newly selected module, the actual \( B_c \) segment has not changed \( (B_c = B_{c-1}) \);
- The distance from the endpoint of the new selected module to the end point of the current \( B_c \) segment is greater than or equal to the distance from the end point of the module facing the new one to the end point of \( B_c \) \( (l^e_c \geq l^e_{c-1}) \);

Visualization of the resulting modular object.

It is important to note that paragraphs 1 and 7 are implemented by means of Unreal Engine 4 using Blueprint, the algorithm for obtaining a modular object is implemented in C++, which allows you to use the presented implementation of the algorithm in conjunction with other software tools.

The results of the algorithm are shown in figure 2.

Figure 2. Example of objects generated using the described algorithm.

Next, we analyze the dependence of the time spent on the reconstruction of three-dimensional models on the number of points in the original data using the presented algorithm for processing an array of information. In addition, the experiment took into account the number of polygons obtained depending on the number of points of the original data (figure 3).

An example of an online course using three-dimensional technologies is shown in figure 3.

Figure 3. A graph of the dependence of the time of building the model on the initial number of points.
In further research work is planned on improving the existing algorithm to generate closed objects, the three-dimensional extension of the model and its possibilities of generation of stacked objects, the next step planned for the research and development of algorithm allowing controlling the variability of modules.

5. Conclusion
In the present work, free software was used, which allowed to develop an algorithm for effective procedural generation of three-dimensional content that meets the conditions of universality and optimality. In the process of research, the developed three-dimensional modular object was presented, and also the model of construction of procedural generation of three-dimensional content is offered.

The proposed model made it possible to create single-tier modular objects taking into account the possibility of combining modules according to the given rules, which ensured the correctness of their construction. This algorithm provided the ability to generate three-dimensional objects in real time, which allowed to quickly making changes in the process of building a three-dimensional scene and thereby optimized the process of generating content. The developed models were tested in the Unreal Engine framework. For a comparison of the approaches were measured and evaluated by such indicators as time spent on the reconstruction and the number of polygons in the resulting model.

Acknowledgments
The research has been supported by the RFBR, according to the research projects No. 17-07-00700 A.

References
[1] Muller P, Wonka P and Haegler S 2006 Procedural Modeling of Buildings ACM Transactions on Graphics (TOG) 25 614-623
[2] Chih-Fan C, Bolas M and Rosenberg E 2017 Rapid creation of photorealistic virtual reality content with consumer depth cameras IEEE Virtual Reality (VR) publisher IEEE (USA: Los Angeles) pp 473-474
[3] Snodgrass S and Ontanon S 2014 A Hierarchical Approach to Generating Maps Using Markov Chains Tenth Artificial Intelligence and Interactive Digital Entertainment Conference 59-65
[4] Liapis A, Yannakakis G and Togelius J 2013 Towards a Generic Method of Evaluating Game Levels Ninth Artificial Intelligence and Interactive Digital Entertainment Conference 30-36
[5] Guzdial M and Riedl M 2016 Game Level Generation from Gameplay Videos Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference 44-50
[6] Hendrikx M, Meijer S, Van Der Velden J and Josup A 2013 Procedural Content Generation for Games: A Survey ACM Transactions on Multimedia Computing, Communications and Applications 9(1) 1-22
[7] Togelius J, Nelson M and Liapis A 2014 Characteristics of Generatable Games Proceedings of the Fifth Workshop on Procedural Content Generation in Games 63-68
[8] Cook M, Colton S and Gow J 2014 Automating Game Design in Three Dimensions Proceedings of the AISB Symposium on AI and Games 20-24
[9] Hoekstra F 2011 Procedural modeling: specialization dissertation Breda University of Applied sciences 16
[10] Burbank D and Hoberman S 2011 Data Modeling Made Simple with CA ERwin Data Modeler R8 Technics Publications 1-45