スクリプトを用いた建築作品の分析とその手法

― 「東京計画 1960」のオフィス棟を事例として ―

○水谷　晃啓*1　八束はじめ*2

キーワード：形態分析　スクリプティング　アルゴリズム　コンピュータ・グラフィックス

1. はじめに

これまでの建築作品の分析・研究は、文章と作図によって検証結果を示すか、復元データから作成したCG画像を用いてそれらを示すものが中心で、分析における操作方法や手順が明確に示されるものは少ない。本論はこれまでの手法だけでは明確に示すことが難しかった、分析の操作やプロセスを、スクリプト言語によって記述することを試みた。また、そのアルゴリズムから、より客観的な情報として分析結果が読み取れることを指摘している。現在、建築業界ではCADによる作図や、グラフィックソフトによるCGを用いたプレゼンテーションが一般的となっているが、使用されるアプリケーションの多くはスクリプト機能を備えている。アプリケーションの動作内容を制御するスクリプトを用いれば、ベースとなるアプリケーションの機能を効果的に活用することができる。本論はこうしたスクリプトの利点を生かし、分析結果の記述を行った。ここでは建築分野で広く利用されるオートデスク社の3ds Maxのスクリプト言語であるmaxscriptを用いている。

2.「東京計画 1960」について

「東京計画 1960」は丹下健三研究室のメンバーによって1961年に発表された。この東京湾上に展開される巨大な計画は、ヴィジョナリーなプロジェクトであるとみなされるが、必ずしもそうとはいいきれない。計画の背後には50年代を通して行われた、東京に対する綿密なリサーチ活動があったからである。「東京計画1960」はデザインのみならず、丹下研のリサーチ研究においてもマニフェスト的なプロジェクトであった。それを示すように、発表されたプロジェクトノートの前半には綿密なリサーチがまとめられ、後半にはデザインがまとめられている。リサーチセクションでは、統計データなどを用いて東京の都市構造が抱える問題や今後の都市のあるべき姿が示される。デザインセクションでは、サイクル・トランスポーテーションと呼ばれる交通システムを持つ都市軸と、都市軸上に配置されるオフィスエリア、海上に浮かべられる住宅エリアの3つが提案されている。それぞれのデザインは東京の人口が、20年後に1500万人になるという予測に基づいて行われており、人口過密の状態をどのように解決するかということが、この計画の主な目的となっている。

3. 丹下モデュールと「東京計画 1960」

丹下モデュールはコルビュジェのモデュロールを基礎とし、丹下およびそのスタッフ達によって模索された寸法体系のことである。この寸法体系は建築・都市を１つの寸法体系でコーディネイトするためのツールとして考案された。広島平和記念公園から使用が開始され、外務省コンペや東京都庁舎でも使われたが、初期の丹下モデュールは工業製品の規格寸法と合っておらず、建物全体の尺度を統一することはできなかった。この問題を踏まえ300 : 600 : 900 : 1500 : …というモデュロールが採用された香川県庁舎では、900割のガラス部材が活用できるなど、ディテール部分を含む、建物全体の尺度の統一に成功する。その後、100 : 165 : 265 : 430 : 695 : 1125 : 1820 : …という日本の規格になじむ伝統的な寸法を含むモデュールが墨記念会館において考案された。墨記念会館（1957年）以降のプロジェクトは1820を含む寸法体系でデザインが行われているため、広島平和記念公園から始まった寸法体系の模索は、この1820モデュールに落ち着いたといってよい。

3DCADを用いた「東京計画1960」（1961年）のCG復元作業を通して、1820モデュールが使用されていることが明らかとなっている。高松一の宮団地の発表に際して、丹下モデュールからなるグリットの上に都市計画が並べられた図表を発表されたが、これはスケールの問題が扱われると同時に、広場のデザインの指標となるように考えられたもので「東京計画 1960」の配置計画においても扱われた課題であった。また丹下モデュールは、スタッフ全員が使用を義務付けられ、丹下研の設計活動においてなくてはならない共通言語でもあった。「東京計画 1960」の分析・CG復元において、丹下モデュールは単に寸法を補うにとどまらず、重要な判断基準となった。

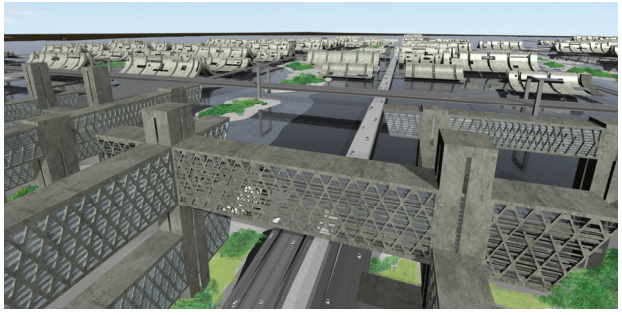

図1.図面と模型写真から行った「東京計画1960」のCG復元画像
この復元作業は計画の主担当であった神谷氏のアドバイスのもと行った。

4. オフィス棟とそのシステムについて

オフィスエリアを担当したのは磯崎新で、垂直なコアと水平なスラブによるデザインは、前年に発表された「新宿計画（淀橋浄水場跡地開発計画）」において考えられたアイデアであった。エレベータや設備などが納められる交通やインフラを支えるコアと、コア間にブリッジ状に架けられたスラブによって、3 次元的に空間がネットワーク化されている。このアイデアは後の「築地計画」においても応用され「山梨文化会館」において実現しており、ジョイント・コア・システムと呼ばれる手法へと発展した。図面から割り出される寸法を追っていくと、丹下モデュールによって各要素の配置やデザインが行われていることが確かめられた。図面では 10 層と 20 層のブリッジ･ユニット（以下:ブリッジ）が噛み合うように組まれているが、模型とパースからブリッジの高さ寸法の組み合わされていることが分かった。これらの図面と模型、パースの分析より得たオフィス棟の構成システムと作品分析結果を以下のようにまとめる。

Analysis 1：コアの配置ルール

任意のコア数が 4 行 5 列のグリットの上に、ランダムに配置されていることが模型から確認できる。

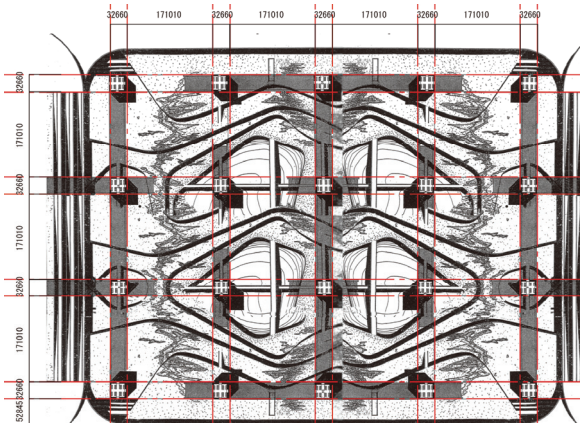

図 2.図面よりコア幅とスパンの寸法割出し

コア幅の寸法は 32660mm×32660mm 、コアの高さ 268990mm、コア間のスパンは 171010mm である。

Analysis 2：ブリッジの種類

10 層と 20 層の 2 タイプが 1・2 スパンに架け渡されるので 4 種類あることが図面と模型から確認できる。

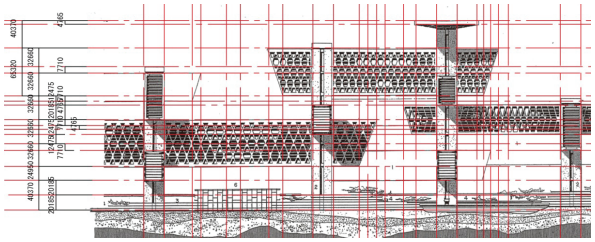

図 3.図面より 10 層と 20 層のブリッジ高さ割出し

ブリッジの高さは 10 層が 40370mm、20 層が 65320mm である。

Analysis 3：ブリッジの配置ルール

平面方向はグリット状、断面方向は 2 つの寸法の組み合わせによって配置されていることがパースより確認できる。

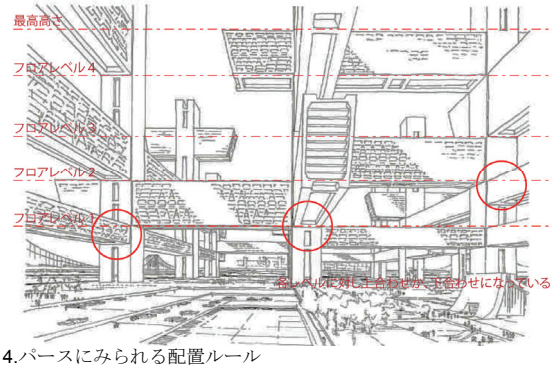

図 4.パースにみられる配置ルール

20 層(mm)	65320	130640	195960	
10 層(mm)	24950	90270	155590	220910

表 1.各ブリッジ下端位置の寸法

20 層のブリッジが積みあがってできるフロアレベルに対して、10 層のブリッジが下端合わせか上端合わせで配置されていることがわかった。

Analysis 4：形態とコンポジション

斜めにカットされたブリッジが 3 次元的にネットワーク化され、ランダムな配置パターンが与えられている。

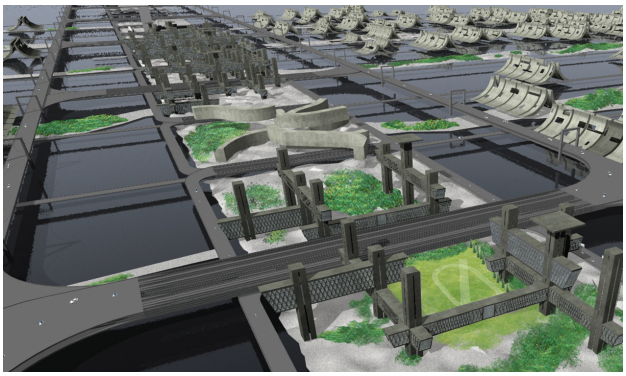

図 5.ブリッジの斜めの形態とランダムなコンポジション

5. スクリプト言語による分析結果（オフィス棟）の記述

ここでは分析によって得た寸法やルールをアルゴリズム化しスクリプトによって記述していく。寸法などの情報や操作をどのようにアルゴリズム化しスクリプティングしたか、全て説明することは紙面の都合上難しいが、maxscript によるコードを用いて可能なかぎり詳しく論じていく。分析から導き出した配置ルールが関数によって定義され、その関数のパラメータとして図面割出しから得た寸法が与えられるというのが、全てに共通する基本操作となっている。前節の分析順序に沿って、主要な寸法や配置ルールがどのようにスクリプトによって定義されているか説明するが、コードの中で Analysis 2 と Analysis 3 が同時に処理されるため、１つのセクションとして扱った。

Scripting 1：コアの配置ルール

```
--関数１：コアの配置位置を与える関数
fn generatepoint posz p_array = (
    --座標をパラメータ（寸法）から求める
    for i = 0 to 4 do (
        posy =i*(171010 +16330)
        for ii = 0 to 3 do(
            p=[ii*(171010 +16330), posy , posz]
            append p_array p
        )
    )
    return p_array
)

--関数２：任意のコア数をランダムに配置する為の関数
fn generatecore base c_no =(
    cores=#()   --作成したコアを格納するための配列
    delList=#() --ランダムに選択したコアを格納するための配列
    --すべての配置にコアを作成し配列に格納する
    for i = 1 to 20 do(
        c = box   length:32660 width:32660 height:268990 wirecolor:white
        c.pos = [base[i][1] , base[i][2] , base[i][3]]
        append cores c
        c.name = "core"+(i as string)
    )
    --選別するコアをランダムに選択し配列に格納する
    for i = 1 to 20-c_no do(
        do(
            done = true
            del = cores[(random 1 20)]
            if (findItem delList del) != 0 do continue;
append delList del
        )
        while not done
    )
    delete delList   --選別され削除リストに格納されたコアを削除する
    return cores
)

PointArray=#() --求められたコアの配置位置(座標)を格納するための配列
generatepoint 0 PointArray --関数１より xy 平面上にコアの配置位置を関数より求める
generatecore PointArray 10 --関数２より任意の数(今回は 10 本)のコア配置を求める
```

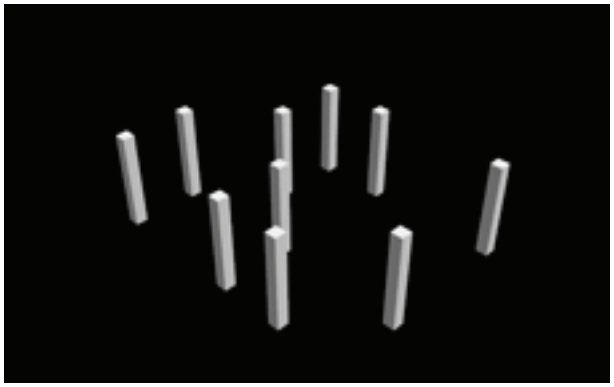

図 6.スクリプトが実行されコアがランダムに配置された

Scripting 2 & 3：ブリッジの種類＋配置ルール

```
--図面の割出から得た寸法をそれぞれ配列に格納する
Hi=#(65320,130640,195960) --高い方（20 層）のブリッジの配置位置（下端の高さ）
Lo=#(24950,90270,155590,220910) --低い方（10 層）のブリッジの配置位置（下端の高さ）
--関数３：高い方（20 層）のブリッジを配置するための関数
fn officeslab1 obj =(
    for i in obj do(
        --コア間の距離が２スパン以内にあるときに実行するように条件づける
        for ii in obj do if (i != ii) and ((distance i ii) <= 374680) then(
            case of(
                --X 方向に直線上に並んだコアが選ばれたときに実行するよう条件づける
                (i.pos.x == ii.pos.x): (
                    c=box   length:((distance i ii)+138350) width:32660 height:65320
                    --配列 Hi に格納された寸法かランダムに配置を求める
                    c.pos=[i.pos.x , (i.pos.y+ii.pos.y)/2 , Hi[(random 1 3)]]
                    c.name=uniquename "office"
                )
                --Y 方向に直線上に並んだコアが選ばれたときに実行するよう条件づける
                (i.pos.y == ii.pos.y): (
                    c=box   length:32660 width:((distance i ii)+138350)   height:65320
                    --配列 Hi に格納された寸法かランダムに配置を求める
                    c.pos=[ (i.pos.x+ii.pos.x)/2 , i.pos.y , Hi[(random 1 3)]]
                    c.name=uniquename "office"
                )
            )
        )
    )
)
--関数４：低い方（10 層）のブリッジを配置するための関数
fn officeslab2 obj =(
    for i in obj do(
--コア間の距離が２スパン以内にあるときに実行するように条件づける
        for ii in obj do if (i != ii) and ((distance i ii) <= 374680) then(
            case of(
                --X 方向に直線上に並んだコアが選ばれたときに実行するよう条件づける
                (i.pos.x == ii.pos.x): (
                    c=box   length:((distance i ii)+138350) width:32660 height:40370
                    --配列Loに格納された寸法かランダムに配置を求める
                    c.pos=[i.pos.x , (i.pos.y+ii.pos.y)/2 , Lo[(random 1 4)]]
                    c.name=uniquename "office"
                )
                --Y 方向に直線上に並んだコアが選ばれたときに実行するよう条件づける
```

```
                (i.pos.y == ii.pos.y): (
                    c=box   length:32660 width:((distance i
ii)+138350)   height:40370
                    --配列Loに格納された寸法かランダムに
配置を求める
                    c.pos=[ (i.pos.x+ii.pos.x)/2 , i.pos.y ,
Lo[(random 1 4)]]
                    c.name=uniquename "office"
                )
            )
        )
    )
)
cores = $core* --作成されたコアをコレクトする
officeslab1 cores --関数３を用いて高い方（20 層）のブリ
ッジを配置する
officeslab2 cores --関数４を用いて低い方（10 層）のブリ
ッジを配置する
```

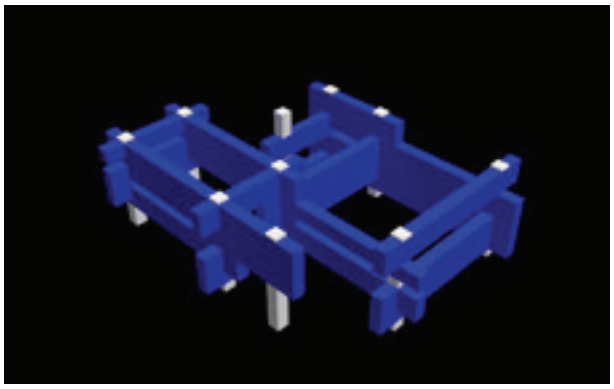

図 7.スクリプトが実行されブリッジが配置された

Scripting 4：形態とコンポジション

```
--関数５：全てのブリッジ配置パターンからランダムに選
別を行うための関数
fn officeposition obj = (
    keepList = #()
    delList   = #()

    do (
        done = true
        for i in obj do (
            --削除リストに入っているものは無視
            if (findItem delList i) != 0 do (
                continue
            )
            for ii in obj do (
                --保持・削除リストに入っているものは無視
                if (findItem keepList ii) != 0 or (findItem
delList ii) != 0 do (
                    continue
                )
                if i == ii do (
                    continue
                )
                if (i.pos.x == ii.pos.x) and abs(i.pos.y -
ii.pos.y) < ((i.length+ii.length)/2) and abs(i.pos.z - ii.pos.z)
<= 65320 do (
                    done = false
                    appendIfUnique delList ii
                )
                if (i.pos.y == ii.pos.y)   and abs(i.pos.x -
ii.pos.x) < ((i.width+ii.width)/2) and abs(i.pos.z - ii.pos.z)
<= 65320 do (
                    done = false
                    appendIfUnique delList ii
                )
            )
            append keepList i
        )
    )while not   done
    delete delList--削除リストに入っているものを削除
)

--関数６：ブリッジに斜めの形態を与えるための関数
fn officetilt obj = (
    for i in obj do(
        --XY 方向に分けて各々ポリゴン操作
        case of (
            (i.length > i.width):(
            t=i.height
            convertToPoly(i)
            i.EditablePoly.SetSelection #Edge #{4}
            move i.selectedEdges [0,t/4,0]
            i.EditablePoly.SetSelection #Edge #{2}
            move i.selectedEdges [0,-t/4,0]
            )               (i.length < i.width):(
            t=i.height
            convertToPoly(i)
            i.EditablePoly.SetSelection #Edge #{1}
            move i.selectedEdges [t/4,0,0]
            i.EditablePoly.SetSelection #Edge #{3}
            move i.selectedEdges [-t/4,0,0]
            )
        )
    )
)

office = $office* --作成されたブリッジをコレクトする
officeposition office --関数５を用いてブリッジを選別する
officetilt office --関数６を用いてブリッジに形態を与える
```

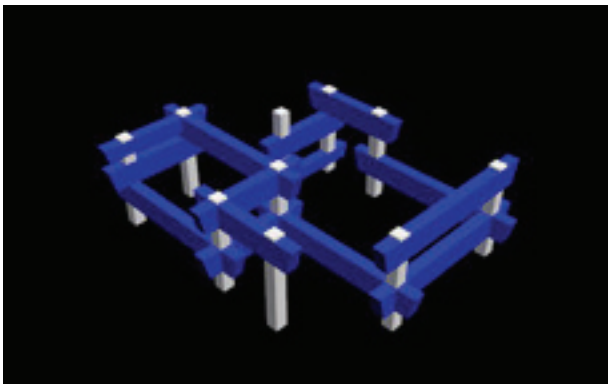

図 8.スクリプトが実行されブリッジが配置された

6. 分析結果とスクリプトの関係性について

4 節で行った分析を 5 節においてスクリプト化したが、本節ではその 2 つの関係について述べる。図面や模型などの資料から読み取った情報と分析結果が、スクリプトによって適正に記述されているか、前節までと同様に段階分けし検証を行っていく。

Verification 1：コアの配置ルール

Analysis 1 において得た 4 行 5 列のコアの配置ルールとコアの幅(32660×32660)と高さ(268990)とコア間のスパン(171010)は、Scripting 1 の関数１と関数２に与えられている。関数１では 171010(スパン)+32660(コア幅)によって芯々寸法を求め、for ループによって 4 行 5 列のコア配列の中心座標を求めた。関数２ではまず for ループ内でコア(32660×32660×268990)を作成し、座標位置に配置するという操作を 20 回繰り返し、全ての座標位置にコアを配置した。次いで配置されたコアをランダムに選択・配列に格納するという操作を、for ループを用いて任意の本数分繰り返し、配列内のコアを削除した。このようにコード内には分析によって得られた寸法と操作が、アルゴリズムとして記述されており、シミュレーションの結果(図 6)で示すように、ランダムなコア配置となることが確かめられた。

Verification 2&3：ブリッジの種類＋配置ルール

Analysis 2 において得たブリッジの種別ルールと寸法（10 層が 40370mm、20 層が 65320mm）と Analysis 3 で得た 3 次元的な配置ルールと寸法(表４)は Scripting 2 & 3 の関数３、関数４に与えられている。関数３では 20 層のブリッジを配置し、関数４では 10 層のブリッジを配置している。表の寸法は配列 Hi=#(65320、130640、195960)と配列 Lo=#(24950、90270、155590、220910)にそれぞれ格納した。どちらの関数も for ループを用いて 2 スパン以内にあるコアの組み合わせを見つけ出し、そのコアの組み合わせが、x 座標が一致するか y 座標が一致していれば配置を行っている。分析によって得たブリッジの配置ルールとするために、配列内(関数３は配列 Hi・関数４は配列 Lo)から、ランダムに寸法値が選択され z 座標に代入されようになっている。シミュレーション結果(図 7)が示すように、関数３と関数４によって配置可能なパターン全てに、4 種類のブリッジが配置されることが確かめられた。

Verification 4：形態とコンポジション

模型写真や Analysis 4 の図 5 からわかるように、ブリッジの粗密はランダムに与えられている。Scripting 4 の関数５はこのランダムな配置パターンを与えるための関数で、関数６はブリッジに斜めの形態を与えるための関数である。関数５では、重なりのあるブリッジを見つけ出し、削除リスト・保持リストとして用意されたそれぞれの配列に振り分けるという操作を、ブリッジの重なりがなくなるまで繰り返し、ブリッジの選別を行った。関数６では選別されたブリッジに対し、斜めの形態を与える操作を行っている。シミュレーション結果(図 8)が示すように、関数５と関数６によってブリッジに斜めの形態とランダムなコンポジションが与えられることが確認できた。

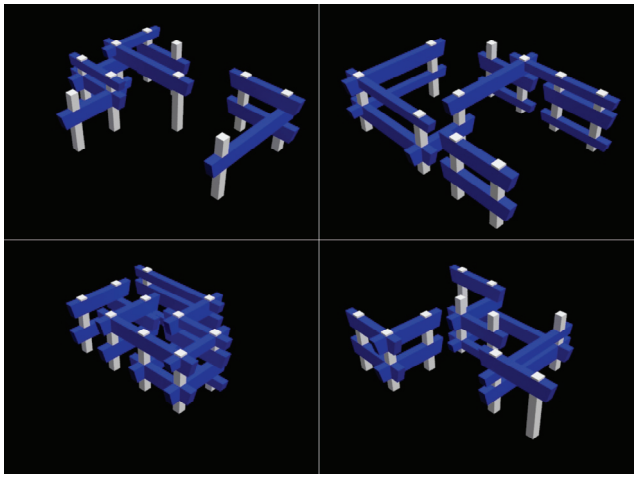

図 9.スクリプトによって得た 4 パターンのバリアント

7. まとめと今後の展開

本論では分析結果をスクリプト言語で記述し、コンピュータ・グラフィックスを用いた建築作品の分析を行った。スクリプトのアルゴリズムが示すように、検証の方法やプロセスを客観的に提示することができた。また、コードを実行すると即時的に検証結果が表示されるため、これまで図面で示されなかった全体像を CG によって視覚的に確認・伝達することが可能となることが分かった。

この手法の特徴である即時的表示と可変性は、都市的構築物の成長過程のシミュレーションにおいて有効な手段ではないかと考えられる。スクリプトは寸法などのパラメータを書き換えるだけで、瞬時に修正が行えるだけでなく、パラメータを変更することで検討の必要があるパターンを、同じアルゴリズムから派生させることができる。そのため、スクリプトはデザインの意図と実際の形態の関係を読み解くことを可能とするツールとなり得るといえる。建築や都市の分析にスクリプトを用いることで、今後より明瞭で客観的な分析を行うことが期待できる。

※この研究は大林財団の支援を受けて行われている。

[参考文献]

1. 丹下健三研究室：新建築，東京計画 1960, 1961.3
2. 丹下健三,藤森照信: 丹下健三, 新建築社, 2002 年
3. Autodesk 3ds Max ビジュアルリファレンス,pp.346~377,株式会社ワークスコーポレーション,2009
4. 豊川斎赫:群像としての丹下研究室,pp.194~208,オーム社,2012
5. 太田利彦,林 昭男:建築とコンピューター,建築雑誌,pp.109~111,1970.02
6. 山田学:コンピュータ・グラフィックスの展開,建築雑誌,p.48,1984.01
7. 長倉威彦:建築家の文法,建築雑誌,pp.30~31,1992.05
8. コスタス•テルジディス,田中浩也(訳):アルゴリズミックアーキテクチュア,彰国社,2010
9. Casey Reas(他),久保田晃弘(監訳)：FORM+CODE -デザイン／アート／建築における、かたちとコード, BNN Inc.,2011

＊1　芝浦工業大学大学院理工学研究科　博士課程後期
＊2　芝浦工業大学工学部建築工学科　教授

Architectural Work Analysis and its Methodology Using Script Language
- Office Building for Tokyo Plan 1960 as an Example -

○Akihiro MIZUTANI*1 Hajime YATSUKA*2

Keywords : Morphological analysis, Scripting, Algorithm, Computer Graphics

Introduction

The previous methodologies for analyzing and researching architectural work have been focused on presenting validation results through words and diagrams, or presenting consideration results through reconstruction CGs. Therefore, few examples exist where operations and procedures of analysis are clearly presented. In this paper, a trial is conducted to describe the operations and the procedures of this analysis in script language. They were hard to be presented clearly, using past methodologies. It is also pointed out that the analysis results could be read as more objective information, using the script language algorithm. Currently, many applications used in the architectural world include script function. Using this function, the base application functionality can be utilized effectively. In the present paper, the analysis results are described utilizing such advantages of the script language. Here, maxscript is used, which is the script language for Autodesk 3ds Max, which is widely used in the architectural field.

Method to Analyze Architectural Works

In the present research, analysis of architectural works is conducted in the following steps.

1. Analyze drawings, models and perspective drawings to establish rules between dimensions and rules for constructing spaces.
2. Break up into steps the operations that are used for analysis, thereby clarifying the operational methods and procedures.
3. Define operational methods at each step as functions using script language.
4. Substitute information obtained through analysis such as dimension into functions as parameter.
5. Execute the script and simulate the result.

In 1 and 2, the architectural works are validated through analyzing drawings or model pictures and are identical to the architectural work analysis methods employed in the past. From 3 on, the script language is used to describe the information obtained in 1 and 2, in order to present the validation methodologies and processes, which were difficult using past methodologies. By conducting scripting in these sequences, the rules between dimensions or the rules for constructing spaces, as well as analysis methodologies and procedures, could be clarified through algorithm. The validation results could also be visually presented by simulation using scripts.

Conclusion

Architectural works using computer graphics were analyzed by describing the analysis result with script language. Using script languages, the validation methodologies and processes which were difficult to be presented using previous methodologies could be presented objectively. It was also discovered that visual communication of results are made possible, since the execution of the code will lead to instantaneous display of computer graphics images. The instantaneous display and adjustability, which are the characteristics of this method is assumed to be a valid means in analyzing architectural works. In the previous methods, there was no choice but to erase and re-establish the dimensions after the model was complete. However with script language, instantaneous revisions could be made by altering the parameters such as dimensions. Script language also makes it possible to generate patterns that need to be studied deriving from the same algorithm by altering parameters and thus it is also valid as a judging tool. It is expected that clear and more objective analysis of works could be made possible by using script language to analyze architectural works.

＊1 Graduate Student, Graduate School of SIT
＊2 Professor, Architecture Building Engineering, SIT.