基于3D动画的软件演化信息可视化

于 涵 王 海 彭 鑫 赵文耘
(复旦大学软件学院 上海 201203) (上海市数据科学重点实验室(复旦大学) 上海 201203)

摘 要 可视化是软件维护和演化研究的一个重要组成部分。一个交互式的3D可视化方案能够将软件演化过程更形象地展示给用户。将软件的演化历史比拟成一座城市的发展过程,用户可以自由地在城市中移动,在把握系统宏观变化趋势的同时,很好地掌控其细节的发展情况。在已有相关工作的基础上,利用unity3D实现了一个原型工具,基本实现了预期目标,能为软件管理者提供方便的可视化信息。

关键词 软件演化,软件维护,可视化,unity3D
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.12.008

Software Evolution Visualization Based on 3D Animation

YU Han WANG Hai PENG Xin ZHAO Wen-yun
(Software School, Fudan University, Shanghai 201203, China)
(Shanghai Key Laboratory of Data Science, Fudan University, Shanghai 201203, China)

Abstract Data visualization is an important research area of modern computer science, especially the software maintenance research. An interactive visualization with 3D animation can show the software evolution history vividly. In our system, software evolution history is compared to the development of a real word city. Users can easily move through the city so that they can view the details of the evolution history as well as the high level trends of software architecture. We developed a prototype tool using unity3D based on some related works. The prototype achieves the goal of providing an easy way to view software maintenance data.

Keywords Software evolution, Software maintenance, Visualization, unity3D

1 引言

在软件尤其是大型软件项目的开发过程中,开发者与管理者的有效沟通非常重要,这决定着需求分析有没有到位、软件开发进度能不能跟上等等。由Shneiderman的实验结果[1]可以看出,数据结构的可视化可以明显提高开发人员对程序的理解。一个软件开发过程的三维动画演示可以直接向管理者反馈开发情况,使管理者更好地控制软件开发过程,保证软件质量,以确保软件开发的高效以及软件维护的顺利。可视化模块在一个合作开发平台上起到一种核心的集成作用,它可以在之后的开发和软件系统的评估中起到关键性作用[2]。

最近几年中,关于数据的可视化研究越来越多,而三维可视化技术也早已是一个热门话题,尽管研究结果很多,但是由于早期的很多研究并没有找到一个合适的喻体来形容软件系统,使得软件工程可视化的发展受到了限制。近几年中最有代表性的软件工程可视化工具就是Code City[3]。如图1所示,将软件比拟成一座城市,城市中的建筑代表文件,通过城市的发展来比拟软件的开发过程,将代码的结构以3D图像的形式显示给开发者。还有一些可视化工具侧重于软件开发过程的监控[4],他们大多是通过动态的监控而非简单地将静态的指标图形化。例如Guillaume等提出用颜色的变化以动画的形式来表现软件开发的演变[5]。然而,这些动态变化的可视化手段比较局限,本文的方法则提供了更为丰富的可视化元素。

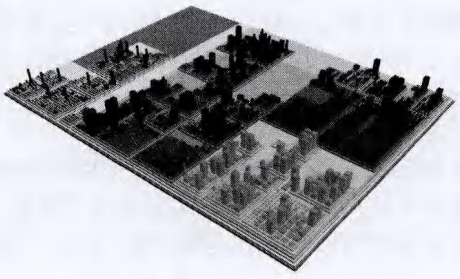

图1 Code City的软件城市

本文提出的可视化手段参照了Code City,将城市作为喻体,利用城市中的各种视觉元素,对软件开发过程中产生的各种度量数据进行映射。在这个理念的基础上,本文利用unity3D技术,将其做成动画。动画相较于静态城市能够更直观

到稿日期:2015-02-24 返修日期:2015-06-17 本文受国家自然科学基金(61370079),国家高技术研究发展计划(863)(2012AA011202)资助。
于 涵(1992-),女,硕士生,主要研究领域为软件工程可视化,E-mail:fishhan@hotmail.com;王 海(1991-),男,硕士,主要研究领域为软件自动化重构、软件工程可视化;彭 鑫(1979-),男,副教授,博士生导师,主要研究领域为软件维护、自适应软件、软件产品线;赵文耘(1964-),男,教授,博士生导师,主要研究领域为软件维护、自适应软件、软件产品线。

地反映出软件系统的变化情况，例如某个类中的方法数量逐渐增多、某个包变得越来越大这些变化情况都能在动画中得以展现。相比于 Code City，管理者能更直观地观察到系统的“腐化”过程，从而及时找出问题所在。

2 可视化设计

本文继承 Code City 的设想，使用城市作为载体。在此基础上，将城市的发展比作软件的开发过程，整个动画就像是一个城市发展史的缩影，从而使开发者和管理者可以直观地感受软件开发的过程和趋势。

2.1 演化度量及视觉维度

在软件演化过程中，会产生大量的度量信息，我们需要筛选部分度量信息作为可视化展示的内容，并将这些内容映射到 3D 图形中。本文以城市作为可视化喻体，通过城市的以下两个特征来对软件演化过程进行可视化：1）城市中建筑物本身的特有属性，如高度、大小等；2）建筑物之间的关系，如是否相邻等。

建筑本身的属性可以用来展示代码单元自身的某些特性，如代码行数、方法数、缺陷密度等。而这些特性又可以被映射到不同的物理特性上，如高度、截面积、颜色等。本文采取表 1 所列的对应关系进行映射。一个方法很多、属性很少的类映射后会呈现一个又高又瘦的建筑；而一个方法少、属性多的则变成了一个“矮胖子”；一个存在很多 bug 的类则是一个深红色的建筑。

建筑之间的关系组成了一个层次结构，而代码本身也具备类似的层级结构。因此，用建筑的布局来映射代码单元间的关系是一种非常合适的选择。在软件系统中，最重要的层级结构是包结构。类和它们所在的包是面向对象结构的主要元素，也是开发者在设计整个软件系统结构时最初考虑的问题。因此，本文考虑包的层级结构，将代表同一个包内的类的建筑摆放在同一个街区，相邻的包在动画中以相邻的街区展现。

表 1 度量映射关系

物理元素	代码元素
建筑	Java 类
建筑高度	类中方法的数量
建筑截面积	类中属性的数量
建筑颜色	类中 bug 的数量

2.2 属性映射

选取度量信息及其对应视觉属性后，本节将讨论以什么样的方式建立映射关系。

2.2.1 模型大小的平衡与规范

最简单的从软件系统的各个度量值到一座城市的各种视觉属性元素间的映射是线性映射。然而，有的类中的方法数相较于其他类可能特别多，此时建筑的高度就需要一个很宽的范围。这种线性映射造成的直接结果就是：当存在超高建筑时，一个平均高度的建筑，就会显得非常矮；而矮的建筑与平均高度的建筑对比则并不明显。此外，如果想要完整地看到全部图像，就需要很大程度地缩小图片，这就将导致大部分中等和小型建筑都无法正常地辨认，观察者的视觉判断也因此可能出现偏差。

另一个更严重的问题是当开发者同时观察这些差距巨大的建筑尺寸时，可能失去了一个城市的感觉，因为从某种程度上来讲，这个城市的图像已经失真了，在格式塔心理学家看来[6]，人对物体的判断是出自于其感觉元素的复合物。所以如果绘制的图像过于失真，就很难达到原本的可视化目标。

为了克服这些缺点，本文通过调整规范高度来解决这个问题，如图 2、图 3 所示。本文以度量为基础，将原始度量数据进行简单分类，把建筑物按照高度的不同分成 5 个类别（非常矮、矮、平均、高、非常高）。对于 Java 系统的每个类中的方法数，本文参考 Code City 的经验拟定阈值为 1（非常低）、4（低）、7（平均）、10（高）、和 15（非常高）[7]。使用这些阈值作为分类的界限，对之前的模型进行平衡，从而得到合适的建筑高度。

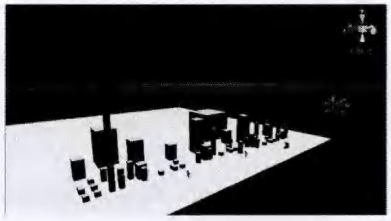

图 2 平衡前失真的城市模型

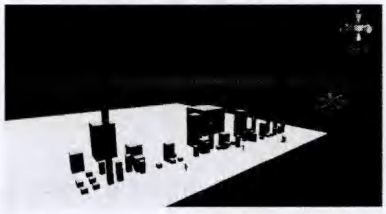

图 3 平衡后的城市模型

2.2.2 三维布局的算法设计

在三维设计中，结构的布局显得尤为重要，一个好的布局往往可以更准确地传递有效信息。本文的三维布局算法充分考虑软件系统内部的内聚耦合问题，用更形象的图像真实地展示软件系统内部的构造。

由于包的嵌套关系，其自底向上地布局在每一个嵌套级别上。对于每个级别的布局，可以归结为一个二维矩形装箱问题，也就是如何在最小的一个矩形面积上将一组矩形组合排列进来。本文采用一个树形算法来进行布局。算法先假设一个足够大的可以容纳所有小矩形的矩形空间（小正方形面积之和的 4 倍大小），然后开始构造一棵二叉树。树的根节点用来描述整个矩形的长与宽，子节点用来描述放入一个矩形后，剩余两个小矩形的长宽。这样整个二叉树的节点就可以记录空间的使用轨迹。当需要定位一个新的矩形时，就开始搜索最适合的叶子结点。为了配合矩形的大小，可以对节点进行递归分割。另外，在每一个区中，每一个矩形的放入顺序是由它的大小决定的，大的矩阵优先放，小的最后放，以防止布局过于碎片化。

算法 1 嵌套布局

输入：所有正方形 Squares

输出：将所有正方形插入完成后的二叉树 Tree

```
FUNCTION Layout(Squares)
1. Sort(Squares);
2. for each Square in Squares
3.     Sizelength+=2 * Square. sidelength;
4. end for
```

```
5. Tree=new tree(Sizelength);
6. for each Square in Squares
7.      Tree=Insert(Tree,Square);
8. end for
9. return Tree
END FUNCTION
```

算法 2 插入正方形

```
输入:二叉树 Tree,正方形 Square
输出:将该正方形插入完成后的二叉树 Tree
FUNCTION Insert(Tree,Square){
1. Fitnode=Tree[0];
2. for each leaf Node in Tree
3. if((Node. width>Square. sidelength
        &&Node. height>Square. sidelength)
        &&(Fitnode. area>Node. area))
4.          Fitnode=Node;
5. end if
6. end for
7. Node1=new node(Fitnode. width-Square. sidelength,
   Square. sidelength);
8. Node2=new node(Fitnode. width,Fitnode. height-Square. side-
   length);
9. Tree. insert(Fitnode,Node1,Node2);
10. return Tree;
END FUNCTION
```

2.3 视角选择

工具采用第一人称视角。第一人称更加符合俯视一座城市发展的上帝视角的体验,相比第三人称更容易让用户把握大局观。另外,工具提供了视角的移动和缩放,快速的视角移动可以提供良好的用户体验,而良好的用户体验可以最大程度地带动人大脑的机动性,让开发者的分析更加准确有效。

3 可视化实现

unity3D 是一个强大的 3D 动画渲染引擎,集成了一套完整的工具集及快速创建 3D 交互内容,并且可以支持多平台发布。我们利用 unit3D 实现了一个原型工具,该工具包含数据获取和动画展示两个模块。

3.1 数据获取

数据获取模块与 git 版本库对接,在版本库中抽取了提交信息、变更内容等原始数据。获取原始数据后,利用抽象语法树对代码进行分析,获取每个 java 类在特定版本所拥有的方法数和属性数。

同时,该模块还与 bugzilla 对接,分析历史记录中某个版本的 bug 数量。从 bug 被报告的时间开始到 bug 被修复为止,该期间的所有提交都被视为包含这个 bug。通过分析一段较长的历史记录,可以较为准确地得到某个版本所含有的 bug 数量。

3.2 动画展示

3.2.1 第一人称视角的实现

在 unity3D 中第一人称视角实现的原理是在动画场景中创建了一个形状如胶囊的对象,并且在胶囊上面绑定了一个摄像机。本文可以通过按键控制这个胶囊体移动,运行动画系统时按 W、S、A、D,视角就会跟随胶囊体移动,同时可以通过鼠标来修改胶囊体的朝向。另外也可以通过鼠标滚轮来实现摄像头的拉近和缩远,这样也是为了可以分别从宏观和微观上感知整个系统的发展变化。

3.2.2 数据库连接

在 unity3D 中数据库的连接使用 C# 实现,在每一个立方体组件中插入 C# 脚本代码,包括连接数据库代码和 select 语句等。之后就可以获得在数据库中的软件系统和版本演化的信息等,从而生成对应的图像。

3.2.3 三维动画实现

获得软件系统的版本数据之后,以 2s 一个版本的速度进行播放,通过 C# 脚本来控制每个部件的生成以及其动画的播放。

4 软件工程演化模型实例

本节以一个简单的螺旋开发模型为例,讲述本文的工具如何体现软件的开发模型,以及对开发过程中风险评估的作用。

在软件开发的过程中,如果遇到一个新的 bug 或者需求,对应的出现 bug 或者需要进行修改的文件或者模块会变成红色。在代码修改的提交过程中,可能其他一些相关的模块也会出现一系列问题。而在进行风险评估的环节,可视化工具可以很好地将模块信息、代码健康情况直观地显现出来。开发者可以基于对动画的视觉感受,采取对应措施。如果一个模块的代码经常性泛红,说明这块代码的"腐化"现象严重,需要及时进行软件重构。然后,在进行工程实现和评审的过程中,之前有问题的代码模块会重新变成绿色,表明这段代码的问题已经解决,等到所有代码全部恢复健康之后,就可以等待进入下一次迭代过程。

管理者可以通过对大片色彩的感知来直观感受软件质量的变化,也可以通过模块的增长和变化来把握软件的宏观发展趋势。这样的设计有助于将抽象的变化数据具体化为易于理解的图像,从而简化管理者对宏观变化的理解过程。当然管理者在把握软件大局的发展趋势的同时,也可以通过缩放来查看细节的变化。通过细节来查看具体是那个模块的变化导致整体的不稳定,从而有目的地进行修正。

在这种不断螺旋迭代开发的过程中,软件的开发轨迹可以很好地展示在管理者面前。同时,在软件开发的过程中,代码的增长速度、开发模式等也可以直观地从可视化动画当中显现出来,从而开发者可以对软件开发的整体开发模型、开发进度、开发质量等做出良好的评估。

5 结论

本文的软件可视化方案的目标就是方便开发管理者快速掌握软件项目的宏观发展趋势以及展示结构分布的细节变化。基于这个可视化方案,我们开发了一个原型工具,实现了可视化信息与用户之间的互动和导航。用户可以根据喜好放大城市中的一个细节,专注于某个特定的地区着重观察,亦或通过放大视角来把握整个城市的宏观发展方向和过程。用户可以看到软件架构的变化情况,从而避免被软件系统本身错综复杂甚至可能是冗余的信息所蒙蔽。

从设计实例上来看,本文的设计基本满足了管理者对于软件演化过程可视化的需求,但还有一些不足之处有待完善。

5.1 当前的不足

本文的设计基本上完成了对一个软件系统的可视化需

求，但是对于一个复杂的软件系统来说，事实上还是有很多的信息并没有完全地表现出来。比如说有关代码的责任人、代码修改的最后提交时间等。

在软件度量信息方面，本文的设计并没有很好地把代码耦合等逻辑关系显示出来，而耦合关系的可视化也是软件演化可视化的重要组成部分，将来可以通过交通路线等来展现不同类之间的耦合关系。

在三维布局上，由于二维迭代布局分布的局限，如果某些代码扩张特别严重，动画或许在视觉感受上会很突兀。

用户的视觉体验会决定用户判断的敏感度，因此界面的美观程度也有待提升，使其更符合城市的感官体验。

5.2 未来的展望

当前本文的城市模型里还有很多元素并没有被加入使用，比如说公路、人，甚至交通情况等。这些元素还可以承载更多的软件开发过程中的情况，比如说人在一座大楼里面工作，可以象征着开发人员在写代码或者修正代码等，这些比喻需要更精巧的思考，以获得最真实的效果和更好的用户体验。

在维度的设计上，还有很多其他可视化元素没有被使用，比如海拔、颜色饱和度、透明度等。这些元素可以用来表达更多的度量信息。例如，透明度可以表示最后更改时间，颜色饱和度可以表示开发人数量，海拔可以用来表述一个类或者包是否是核心算法的类或包等。

另外，布局上也可以突破二维的限制，通过架设立交桥等其他设想来表现更好的逻辑耦合关系连接，让整个设计既能符合人的社会认知，又能更好、更丰富地表现系统内部的各种信息等。

可视化工具已经成为当前软件维护开发的重要组成部分，而将抽象的软件系统结构比拟成用户所熟悉的现实环境的具体形象，可以更好地提升用户体验，让一些抽象的特征更加直观，从而使软件管理者更好地掌握关键信息，快速做出正确的决策。相信随着设计的进一步完善，它一定能为开发人员提供更大的帮助。

结束语 本文通过对软件开发过程进行建模并使用三维动画做演示，将抽象的软件开发过程对应成城市的发展过程，使管理者更好地获得软件开发过程的反馈，从而提高了软件开发效率，确保了软件开发质量。

参 考 文 献

[1] Ben S. Control Flow and Data Structure Documentation: Two Experiments [J]. Communications of ACM, 1982, 25(1): 55-63

[2] Limberger D, Wasty B, Trumper J, et al. Interactive Software Maps for Web-Based Source Code Analysis[C]// Proceedings of the International Web3D Conference, 2013. ACM, 2013, 475(3): 91-98

[3] Wettel R, Lanza M. Visualizing software systems as cities[C]// VISSOFT. 2007. Banff, Ont., 2007: 92-99

[4] Beyer D, Hassan A E. Animated Visualization of Software History using Evolution Storyboards[C]// WCRE 2006. Benevento, Italy, 2006: 199-210

[5] Langelier G, Sahraoui H, Poulin P. Exploring the evolution of software quality with animated visualization[C]// Visual Languages and Human-Centric Computing. 2008: 13-20

[6] Few S. Show me the numbers: Designing Tables and Graphs to Enlighten[M]. Analytics Press, 2004

[7] Wettel R, Lanza M. Program Comprehension through Software Habitability[C]// ICPC. 2007. Banff, Alberta, Canada, 2007: 231-240

(上接第 7 页)

[36] Ramesh A, Srinivasa K, Pramod N. SentenceRank—A graph based approach to summarize text[C]//2014 Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT). IEEE, 2014: 177-182

[37] Baralis E, et al. GraphSum: Discovering correlations among multiple terms for graph-based summarization[J]. Information Sciences, 2013, 249: 96-109

[38] Goyal P, Behera L, McGinnity T M. A Context-Based Word Indexing Model for Document Summarization[J]. IEEE Transactions on Knowledge and Data Engineering, 2013, 25(8): 1693-1705

[39] Cai X Y, Li W J. Ranking Through Clustering: An Integrated Approach to Multi-Document Summarization[J]. IEEE Transactions on Audio Speech and Language Processing, 2013, 21(7): 1424-1433

[40] Zhang Z C, Ge S S, He H S. Mutual-reinforcement document summarization using embedded graph based sentence clustering for storytelling [J]. Information Processing & Management, 2012, 48(4): 767-778

[41] Sankarasubramaniam Y, Ramanathan K, Ghosh S. Text summarization using Wikipedia[J]. Information Processing & Management, 2014, 50(3): 443-461

[42] Otterbacher J, Erkan G, Radev D R. Biased LexRank: Passage retrieval using random walks with question-based priors[J]. Information Processing & Management, 2009, 45(1): 42-54

[43] Hariharan S, Ramkumar T, Srinivasan R. Enhanced graph based approach for multi document summarization[J]. Int. Arab J. Inf. Technol., 2013, 10(4): 334-341

[44] Wang W, et al. Exploring hypergraph-based semi-supervised ranking for query-oriented summarization[J]. Information Sciences, 2013, 237: 271-286

[45] Diaz A, Gervas P. User-model based personalized summarization [J]. Information Processing & Management, 2007, 43(6): 1715-1734

[46] Li S, Wan W, Wang C. TAC 2008 update summarization task of ICL[C]// Proceedings of the Text Analysis Conference (TAC). 2008: 132-139

[47] 刘美玲，等. 动态多文档文摘模型[J]. 软件学报，2012，23(2)：289-298
Liu Mei-ling, et al. Dynamic Multi-Document Summarization Model[J]. Journal of Software, 2012, 23(2): 289-298

[48] Minkov E, Cohen W W. Adaptive graph walk-based similarity measures for parsed text[J]. Natural Language Engineering, 2014, 20(3): 361-397

[49] Lee M C, Chang J W, Hsieh T C. A Grammar-Based Semantic Similarity Algorithm for Natural Language Sentences[J]. Scientific World Journal, 2014(1): 437162

[50] Skabar A, Abdalgader K. Clustering Sentence-Level Text Using a Novel Fuzzy Relational Clustering Algorithm[J]. IEEE Transactions on Knowledge and Data Engineering, 2013, 25(1): 62-75