# Variable Feature Sets in Grid-based Layouts

Kiran Gajanan Javkar
Samsung R&D Institute Bangalore
kiran.javkar@samsung.com

Pravin Kumar Ramesh Mali
Samsung R&D Institute Bangalore
pravin.mali@samsung.com

Ashutosh Kumar
Samsung R&D Institute Bangalore
ashutosh.86@samsung.com

Lokesh Chinnaga
Samsung R&D Institute Bangalore
lokesh.ch@samsung.com

## ABSTRACT
As the digital data continues to grow unbounded, users expect intelligent processing and accurate coverage of all its domains. Advances in display technology have led to a variety of viewing devices, each with its own characteristic sizes and resolutions. Thus, a single layout on different sized devices would degrade the user experience. Moreover, the information shown on display does not need to have same features. So, it becomes very crucial to develop a user interface which can cope up with different sized viewing devices and different data set information. To facilitate the same, we hereby present a novel approach to identify key information from various data sets and display the same on wide range of devices using comparable but different grid-based layouts.

## Categories and Subject Descriptors
H.5.2 [**Information Systems**]: Information Interfaces and Presentation - *User Interfaces*

## General Terms
Algorithms, Design, Human Factors.

## Keywords
Adaptive Layout design, Graphical User Interface, Constrained Based User Interface, Grid based Layout, Responsive Layout design

## 1. INTRODUCTION
The digital data available continues to explode, as various new files and web pages are created continuously. Although the data relevant to a user can be identified, no system has been used which showcases the important information for each of these data items together, in a grid-like structure. Currently available grid-based layouts used for displaying heterogeneous data have fixed predefined parameters which specify the layout format and is, hence, across the devices, irrespective of devices' screen size and resolution. On account of advancement in digital display technologies, a wide range of display devices are made available, each having a different display size and resolution. As it is thus feasible to showcase varied extent of information for a chosen data item, depending on the device used for displaying [1], a single fixed layout across all the devices would degrade the user experience.

In this paper we seek to present a new generalized algorithm to facilitate displaying of most important information for various data items using variable width cells in grid-based layout. Our approach is designed to identify the widths of each data item based on the important features identified for that item as well as the screen properties for the provided device. This ensures that the data items are provided sufficient, yet variable, screen space to display most of the data that can be shown on the given device screen while maintaining the consistency in display [9] using the grid-based layout.

The second section of this paper deals with the related work performed in this field. Third section specifies the dataset used to validate our approach. Fourth section states the algorithm used to generate the required grid-based layout for various categories of data items provided in the dataset while considering the device screen specification used for displaying the resultant layout. Section five shows the grid-based layout generated for a sample dataset. Section six denotes some future work relevant to the proposed approach. Lastly, section seven enlists the references used in formulation of this paper.

## 2. RELATED WORK
The increasing sophistication levels of modern day graphics hardware have resulted in intelligent user interfaces for computer-based graphical communication. Much research in this area, however, has been focused on the automatic synthesis of graphics for either presenting relational information [1] or realistic depictions of 3D objects as in [2, 3]; the automatic layout design of graphical presentations has been relatively unexplored. Development of advanced interfaces for devices with varying sizes essentially requires automating the graphical designing combined with different modal characteristics, like animation, to facilitate the user to communicate in efficient and expressive format. Coherency in output can be achieved by the ability to reflect certain semantic and pragmatic relations specified by presentation planner in order to arrange the visual appearance of a mixture of textual and graphic fragments delivered by mode-specific generators [4]. The coordinated combination of different modalities, like natural language, graphics, animation, becomes an important factor in developing sophisticated intelligent UI providing flexible and efficient information presentation [5, 6, 7].


*Compute 2015,* October 29-31, 2015, Ghaziabad, India
 ISBN 978-1-4503-3650-5/15/10…$15.00
DOI: http://dx.doi.org/10.1145/2835043.2835051

Eastman's work on a Geneml Space Planner that addressed the task of arranging objects, such as furniture, given a set of constraints ([8], chap. III) can be viewed as an early effort in automating layouts. Feiner's GRIDS (Graphical Interface Design System) was constructed as an experimental system for investigating approaches in automatic display layout of text, illustrations, and later virtual input devices. Based on the concept of design grid, their system generates a grid intended for a set of possible displays, based on information about the kinds of objects to be presented, to create the prototype display. This prototype display had regular grid spacing, which meant regular sizes and positions of objects displayed, thereby achieving consistent and coherent look for individual display [9]. The consistent layout irrespective of the device used, however, degraded the user experience. Some other approaches that followed this include use of computer-based grids modelled by a human designer, as found in VIEW [10] for synthesizing graphical object depictions from high-level specifications and by [11] for low-level table layout, whose high-level topology was specified by the user as a matrix.

The last few decades saw sophistication of constraint-satisfaction techniques, growing availability of advanced graphics and the resultant trend inclined towards applying constraint techniques to UI design.

Beach has demonstrated that determining whether an arbitrary set of non-overlapping rectangular objects can fit on a display is NP-complete; same is the case to find the minimum size rectangle in which the objects can be packed. The challenge, hence, worth considering is proposing the set of constraints and evaluation criteria that would allow effective layout generation.

Sketchpad [12], an interactive graphics and constraint satisfaction based system, by Sutherland (MIT, 1963), allowed a user to create complex objects by sketching primitive graphical entities and specifying constraints on them. Majority of these ideas were explored for ThingLab system at Xerox PARC [13] and multiple versions were developed concerning extensions supporting constraint hierarchies, incremental compilation, and graphical facilities for defining new kinds of constraints (e.g. [14, 15]). Other systems and research activities in constraint based graphics include Steele's constraint language [16], Juno [17], IDEAL [18], Magritte [19], Bertrand [20], work by Cohen et al. on constraint-based tiled windows [21].

Charles Jacob's work to bring online publication design at par with paper-based proposed automatically adapting page layouts, using grid-based design principles, for appealing page layouts matching their displays. This was majorly suited for text based information [22]. A large number of interface-design systems, predominantly based on graphical editor, have been developed over the past few years, mainly targeted for a more efficient and convenient interface design. The constraints, here, provide a means for stating layout requirements and are typically complicated and difficult to calculate; e.g. Peridot system, which deduces constraints automatically as user demonstrates desired behavior [23].

It can be seen that the previous efforts enlisted do not take into consideration the type features and characteristics that can be exhibited by the objects to be displayed in the grid, and the possible distinguished attributes of the same. To address the shortcomings of the current user experience on devices varying in size dimensions and pixel densities and to display varied type of information on every single device, we, hereby, propose an adaptive user interface layout algorithm to display information of objects via info-cards on a variety of devices while retaining the ability to produce quality grid-based layouts.

# 3. DATASET

The dataset for our algorithm includes all such data elements which have identifiable set of features and can be shown in a grid format. Examples of items that can constitute the dataset include files and folders that can be shown as thumbnails, such as images, videos, document files, compressed file folders; web URLs as well as customized items with explicitly specified feature set. Feature set for items of the type files and folder could be file type, folder type, read/write permissions, creation date, size etc. Feature set for web URLs could be various tags mentioned and categorization of URLs based on availability and content of various default as well as customized tags.

# 4. ALGORITHM

Our algorithm relies on identification of maximum number of Info-Cards that are to be shown on a given device screen. In addition to this, the maximum number of features for the given set of Info-cards is also identified. The feature set for an Info-Card can be viewed as a set of relevant features for the card data that are deemed considerable for showing on a UX display. A global feature set will be maintained, which will comprise of the maximum permissible features that can be identified across the entire Info-Card dataset. The feature set to be identified for each Info-Card will, hence, be a subset of this global feature set. The value obtained for maximum number of features for Info-Card is, hence, expected to be small, typically less than 10. Based on these two values, the screen is virtually divided into multiple equal spaced columns.

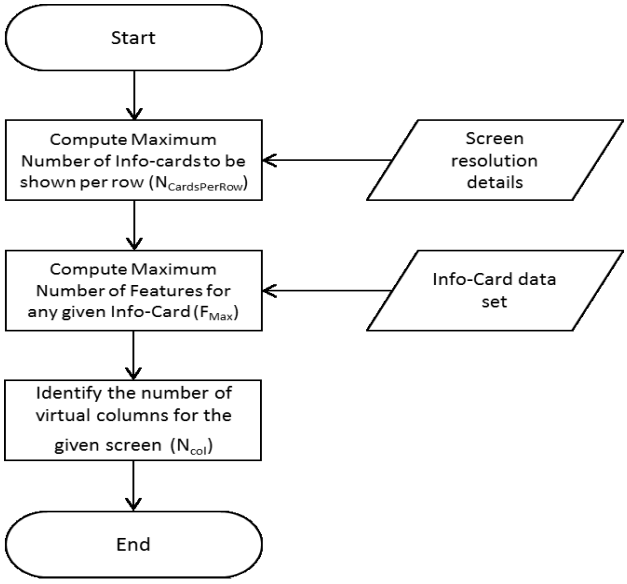


**Figure 1. Flow for Identification of number of columns and maximum cards per row for a given device screen**

Partitioning the screen virtually into a number of columns, as identified using the algorithm shown in Fig. 1, serves as a base for positioning the Info-cards into variable sizes, depending upon the number of features present in that card and other adjoining cards. Our algorithm will operate on stream of Info-Card data elements, which will constitute its input dataset, and will suggest the best

fitting column spanning widths for these elements in a sequential format.

## 4.1 Info-Card

As aforementioned, each Info-Card will have a set of features, where the count of the same for any card will be at most $F_{Max}$. The features identified for an Info-Card can be completely different from some other Info-Card. The count of the features identified for each card will act as the best suited column spanning width or the default width for that card.

Each of the features identified for different Info-Cards will constitute a global feature set and will have a relative priority amongst them. This relative priority can be viewed as an integer value assigned to a feature such that bigger the number assigned, higher would be its priority.

## 4.2 Algorithm to Identify Best Fitting Column Spanning Width

Once we have the maximum number of Info-Cards to be shown in a row ($N_{CardsPerRow}$), we use this algorithm to identify the best fitting column widths for each of these cards. The algorithm will operate on at most $N_{CardsPerRow}$ Info-Cards simultaneously. The pseudo code for this algorithm is as follows:

Input:

*info_card* ← stream of Info-Card data elements

$N_{CardsPerRow}$ ← Maximum number of Info-Cards that can be shown in a row

$N_{col}$ ← Number of virtual columns identified for the given device screen

Output:

Best fitting column spanning widths for *info_card* data elements in sequential manner

Procedure:

1. *Offset* ← a fraction of $N_{col}$, required to ascertain whether a card is to shown in current row or next row.
2. While *info_card* not empty
   a. If at least (2*$N_{CardsPerRow}$) items are available in *info_card*, consider the initial $N_{CardsPerRow}$ number of items, else consider all items available
   b. *CurrentRowCards* ← empty
   c. Add first item available from *info_card* to *CurrentRowCards*
   d. *TotalCurrentDefaultSize* := Default width of first item available from *info_card*
   e. Remove the first item from *info_card*
   f. While *TotalCurrentDefaultSize* ≤ $N_{col}$ + *Offset*
      i. If Default width of the current first item in *info_card* is less than or equal to ($N_{col}$ + *Offset*) - *TotalCurrentDefaultSize*
         1. Add this item to *CurrentRowCards*
         2. Update *TotalCurrentDefaultSize* by adding Default width of this item to the existing value
         3. Remove this item from *info_card*
   g. While *TotalCurrentDefaultSize* not equal to $N_{col}$
      i. If *TotalCurrentDefaultSize* is less than $N_{col}$
         1. Increase the size assigned to each item in *CurrentRowCards* by adding a value of ($N_{col}$ - *TotalCurrentDefaultSize*) / *CurrentRowCards* to their respective Default widths
      ii. Else
         1. While *TotalCurrentDefaultSize* not equal to $N_{col}$
            a. Identify the item from *CurrentRowCards* which has the lowest priority feature as specified in global feature set such that its current assigned width is more than 1
            b. Exclude this feature from this item for further algorithmic considerations
            c. Reduce this item's assigned width and update *TotalCurrentDefaultSize*

The above algorithm can be also provisioned with user-defined values for both $N_{col}$ and $N_{CardsPerRow}$, in addition to their screen-based computation mentioned before. This would facilitate explicit specification of the maximum number of Info-Cards to be shown alongside specifying the number of virtual columns, thereby provisioning to display the grid-based layout which supports both vertical and horizontal scrolling, if required.


**Figure 2. Grid-Based Layout for six Info-Cards using our algorithm ($N_{col}$ = 4 and $N_{CardsPerRow}$ = 2)**

## 5. RESULTS

We applied our algorithm on a device screen for which $N_{col}$ was identified as 4 and $N_{CardsPerRow}$ was identified as 2. Our dataset included Bird Images, TV Series, Sports URL, Maps, and Accessory URL. The Global Feature set comprised of all the identifiable feature such as TV series name, TV series genre, TV Series Aired, Location's latitude and longitude, distance between any two locations, Cricket match scores and related details, Image resolution, accessory price, accessory company name and other such details. High priority features could be Brand Name, Price, match scores, image, etc.

Low priority features could be image size, image resolution, length of description, and other trivial and not so significant URL details.

Figure 2 shows the grid-based layout generated for the above mentioned Info-card data items using our proposed algorithm.

Figure 3 shows the grid based layout generated using our algorithm for the same dataset, which was used to generate the layout shown in Figure 2, but with a different device screen for which $N_{col}$ was identified as 6 and $N_{CardsPerRow}$ was identified as 3.


**Figure 3. Grid-Based Layout for six Info-Cards using our algorithm ($N_{col}$ = 6 and $N_{CardsPerRow}$ = 3)**

## 6. FUTURE WORK

Our proposed algorithm specifies a standard height for populating all the Info-Cards in the dataset; their width however, varies, as discussed before. However, it might also be required to populate the cards with height and width, both variables. This is an imminent enhancement for the proposed algorithm, which would be targeted in near future.

There may be some data elements which have width range constraints; that is, they may have explicitly specified maximum and minimum width that can be assigned to respective data items. Future work would also be targeted to incorporate such constraints.

## 7. ACKNOWLEDGMENTS

We would like to express my sincere gratitude to our senior team members and advisors, Jeelani Basha Manikindi, Sainath Kasi and Joy Bose, for their continuous support for our study and related research, for their patience, motivation, and immense knowledge.

## 8. REFERENCES

[1] J.D. Mackinlay. Automatic Design of Graphical Presentations. PhD thesis, Dept. of Computer Science, Stanford University, Stanford, CA, 1985.

[2] D. Seligmann and S. Feiner. Automated generation of intent-based 3d illustrations. Computer Graphics, 25(3), July 1991.

[3] T. Rist and E. Andre. From presentation tasks to pictures including depictions of 3D objects: Towards an approach to automatic graphics design. German Research Center for Artificial Intelligence, DFI, 1992

[4] Winfried Graf , Constraint-Based Graphical Layout of Multimodal Presentations, German Research Center for Artificial Intelligence (DFKI) Stuhlsatzenhausweg 3, W-6600 Saarbrucken 11, Germany, January 1992

[5] J. Sullivan and S. Tyler, editors. Intelligent User Interfaces. Frontier Series. ACM Press, New York, NY, 1991.

[6] W. Wahlster, E. Andre, S. Bandyopadhyay, W. Graf, and T. Rist. WIP: The coordinated generation of multimodal presentations from a common representation. In Ortony et al. Also DFKI Research Report RR-91-08, Germany, 1992

[7] M. Maybury, editor. Intelligent Multimedia Inte1jaces. AAAI Press, Menlo Park, CA, 1992. Forthcoming.

[8] A. Barr and E. Feigenbaum, editors. The Handbook of Artificial Intelligence, Vol. 1. Pitman, London, 1981.

[9] S. Feiner. A grid-based approach to automating display layout. In Proceedings of the Graphics Interface '88, pages 192- 197. Morgan Kaufmann, Los Altos, CA, June 1988.

[10] M. Friedell. Automatic synthesis of graphical object descriptions. Comput er Graphics, 18(3):53-62, 1984.

[11] R. Beach. Setting Tables and Illustrations with Style. PhD thesis, Dept. of Computer Science, University of Waterloo, Ontario, 1985.

[12] I. Sutherland. Sktechpad: A man-machine graphical communica.tion system. In IFIPS Proceedings of the Spring Joint Computer Confe1'cnce, pa.ges 329- 345, 1963.

[13] A. Borning. The programming language aspects of ThingLab, a constraintoriented simulation laboratory. ACM Transactions on Programming Languages and Systems, 3(4):353-387, October 1981.

[14] A. Borning, R. Duisberg, B. Freeman-Benson, A. Kramer, and M. Woolf. Constraint hierarchies. In Proceedings oj OOPSLA '87, pages 48- 60, October 1987.

[15] B. Freeman-Benson, J. Maloney, and A. Borning. An incremental constraint solver. Communications of the ACM, 33(1):54- 63, 1990.

[16] G.J. Sussman and G.L. Steele. Constraints - a language for expressing almost-hierachical descriptions. Artificial Int elligence, 14(1): 1- 39, 1980.

[17] G. Nelson. Juno, a constraint-based graphics system. Pmceedings of th e SIGGRAPH '85, 19(3):235- 243, 1985.

[18] C.J. van Wyk. A high-level language for specifying pictures. ACM 1hmsactions on Graphics, 1(2):163-182, 1982.

[19] J. Gosling. Algebraic Constraints. PhD thesis, Dept. of Computer Science. Carnegie Mellon University, 1983.

[20] W. LeIer, editor. Constraint Programming Languages: Their Specification and Generation. Addison-Wesley, Reading, MA, 1988.

[21] E. Cohen, E. Smith, and 1. Iverson. Constraint-based tiled windows. IEEE Computer Graphics and Applications, 6(5):35-45, 1986.

[22] Chuck Jacobs, Wilmot Li, Evan Schrier, David Bargeron, and David Salesin, Adaptive Document Layout, Aug. 2004.

[23] B. Myers. Using AI techniques to create user interfaces by example. In Sullivan and Tyler, pages 385-402. Peridot, 1991