

モデルとビューを分離したプレゼンテーション用ビジュアルエイドシステム

大坪五郎*

概要. プレゼンテーションを視覚的に補助するビジュアルエイドは、通常「スライド」として作成される。しかし「スライド」にはプレゼンテーションのストーリーとして喋る内容と、その中で強調するため画面に表示すべき部分が混在している。本来プレゼンテーションで一番重要なのはストーリーでありそれを「モデル」と捉えれば、ビジュアルエイドとして人の目に触れる部分の記述を「ビュー」として分離することが必要なのではないかと考えた。そうした問題意識に基づいたプレゼンテーション用ビジュアルエイドシステムを開発した。まず平易な文章でプレゼンテーションのストーリーをモデルとして作成する。その上でストーリー中の最小限必要な部分を選択し、画面に表示する際の仕様を簡易なビジュアルプログラミング言語でビューとして定義する。これによりプレゼンテーションのストーリーをスライドという形式にとらわれることなく検討することができる。またストーリーを語る際に情報の表示タイミング、方法を細かく制御することが可能となる。本システムの開発及び実際に使用した結果について述べる。

1 はじめに

「明日のプレゼンテーションのスライド全然できてないよ」のような言葉を聞くことは多い。この言葉の背景には、多くの人がプレゼンテーションの準備をすることを「スライドを作る」とことと等価と考えているためと考えられる。しかしながらこの「スライド作成＝プレゼンテーション準備」という想定は正しくない。プレゼンテーションで最初に検討すべきはプレゼンテーションのストーリーであり[6]、喋る内容である。自分がプレゼンテーションで訴えたい内容はなんなのか。それを聴衆にわかってもらうためにはどのような項目をどのように述べればよいかをまず十分に検討し、その上でストーリーを補助するために画面上に情報を表示するというのが本来の姿—ビジュアルエイドという言葉の意味—であったはずだ。

現状広く使われているプレゼンテーションソフトウェアはいずれもスライドをベースにしている。これはおそらく過去に使われていた物理的実態を持った「スライド」を置き換える目的からスタートしたためと考えられる。しかしスライドをベースとした方法がプレゼンテーション準備をする上で最適な方法であるとは限らない。

スライドをベースにプレゼンテーションの準備を行うことの一番のメリットは画面に映し出す情報を直接構築・編集できるため、手間が省ける点にあると考えられる。しかしながら、プレゼンテーションにおいて最も重要なプレゼンテーションのストーリーは連続的な自然文で記述・検討されるべきである。しかしスライド（及び補助としてノート）の集合体

でそうしたストーリーを記述するのは容易ではない。

さらには、昨今プレゼンテーション用スライドは、単にプレゼンテーションの最中に用いられるだけでなく、プレゼンテーション終了後のハンドアウトとしても使われることが多い。しかしながらこの二つの機能を一組のスライドで賄うのには問題がある。本来プレゼンテーションは、プレゼンターが聴衆に語りかけるものである。画面に映し出すのはその語りかけを支援するための最小限の情報であるべき。逆にハンドアウトとして渡す際には、それだけで受取手がほとんどの情報を得られることが期待される。前者の目的にフォーカスすれば画面上の情報は少なくなり、ハンドアウトとして渡されても意味が取れないことになる。後者の目的にフォーカスすれば、表示される文字は必要以上に多くなり、プレゼンテーション時において聴衆は退屈な「画面上の文字の読み上げ」に付き合わされることになる。こうした問題はプレゼンテーションの補助を行うアプリケーションはどのような機能を持つべきか、という根本的な点に立ち戻ることなく従来のスライドという枠組みの中で問題を解決しようとしていることから生じるものではないだろうか。

こうした根本的な問題意識に基づき、プレゼンテーションを支援するビジュアルエイドの表示制御はどのように行うべきかという考察に基づいたプレゼンテーション用ビジュアルエイドシステムを開発した。

2 プレゼンテーション用ビジュアルエイドシステムの開発

2.1 想定するプレゼンテーションスタイル

プレゼンテーションを行う際に、ストーリー並びに発する文言をどの程度事前に用意するかについては様々な議論がある。ここではUrban[5]の定義に基づき議論を行う。Urbanによればプレゼンテーションのスタイルは以下の3パターンに分かれる。

1. 構成、文言とも全く用意せず、その場の即興で喋る。
2. 構成だけ決めておき、文言はその場で考える。
3. 文言まで用意しておく。このパターンは以下の3サブパターンに分類される。
 - (a) 原稿を読み上げる。
 - (b) 原稿をなんとか見なくても良いレベルまで暗記する。
 - (c) 意識を逸らしても言葉がでてくるレベルまで暗記する。

これらのパターンにはそれぞれメリット、デメリットがありどのパターンを採用するかはプレゼンターの置かれた状況、それぞれのパターンへの習熟度合いによって変わってくる。日常目にするプレゼンテーションはその多くがパターン2に属すると推定される。プレゼンテーションの構成ならびに大まかに喋る内容は全てスライド上に記述されており、それに従いプレゼンターが半ば読み上げ、半ば即興で文言を並べるスタイルである。そして現在のスライドを主体としたソフトウェアはこのパターン2のプレゼンテーションスタイルに非常に適合している。しかしながら話そうとする項目をスライド上に並べるだけで「スライドが完成」してしまうため、本来最も慎重に時間をかけて行うべきプレゼンテーションストーリーの検討がおろそかになる、といった問題がある。

またUrbanの推定によればTEDで行われるプレゼンテーションの9割はパターン3-cに分類される。パターン3では、詳細にどのような文言を使うかにまで踏み込んで準備を行う。既存のプレゼンテーションソフトウェアではスライドごとにノートを文字の形で記述することはできるが、それらはスライド単位で細切れになっており、流れ全体を文章の形で俯瞰し、検討することはできない。

パターン1は最も失敗の可能性が高く、失敗をできるだけ避けたいプレゼンテーションには向かないと考えられる。そのために本論文においてはパターン2, パターン3-a~3-cのプレゼンテーションを支援することを前提として検討を行う。

2.2 本システム開発前の実運用

著者は以前より、プレゼンテーションのビジュアルエイドは、画面上の詳細なレイアウトの定義より画面に表示されるタイミングのコントロールを主眼とするべきであり、スライドではなくプログラムとして記述すべきである、という主張を持っていた。この構想にしたがって開発したのがGozen[9]である。このシステムは独自のプレゼンテーション記述用言語を用いてビジュアルエイドの表示仕様を記述するものであった。筆者はこのシステムを持ちいて合計4回社外向けプレゼンテーションを実施した。その結果当初の目論見通りビジュアルエイドの記述が直感的になり、かつプレゼンテーションに合わせて表示タイミングを細かく制御できるというメリットを確認できた。それと同時に以下のような問題点も明らかとなった。

1. 独自プログラミング言語を使用することによる、信頼性の低さ。
2. エディタのコード補完機能がサポートされないことによる開発効率の低さ。
3. プログラムのモジュール化が十分でなく、長いプレゼンテーションになると全体の把握/編集が困難になる。
4. プレゼンテーションのストーリーをプログラム中のコメントとして残すことは可能だが、プログラムの編集・更新とともにコメントとして残したストーリーと表示内容が乖離する。

これらの問題のうち、1-3番目に対処することを目的として、システムの機能はGozenと同等とし、Mac上で稼働するSwift言語を使いGUIを持つプログラムとしてプレゼンテーションの記述を行うことを試みた。作成したシステムを用いてWISS2016を含め3度社外向けプレゼンテーションを実施した。信頼性の高いSwiftを用いることでプレゼンテーション時の心理的負担は軽減された。またMac上のツールとして開発された各種プログラミング支援機能を利用することで開発効率が向上した。さらに複数の画面表示要素を一つのモジュールとしてまとめ、それらを並べることでプレゼンテーションを構成するといったやり方の利点を実感することができた。しかしながら4番目の「プレゼンテーションのストーリーと画面表示内容が分離する」という問題に対してはこの方式では解決が難しい。他に明らかになった問題としては以下のものが挙げられる。

- 画面表示がどのようなになるかは一旦プログラムをコンパイルした後でしかわからない。このため修正に時間がかかる。
- プログラムの記述量が多く、簡単なプレゼンテーション用のビジュアルエイドであっても作成に多くの時間を要する。

2.3 システム設計方針

前項で述べた経験結果を踏まえ、以下に示す設計方針をたてた。

- ビジュアルエイドをモデルとビューに分離する。モデルとはプレゼンテーションのストーリーであり、自然言語で記述する。その中から画面に表示すべき部分を指定し、表示仕様を示すビューを設定することによりビジュアルエイドを構成する。
- ビューの記述は、編集が容易なビジュアルプログラミング言語により行う。
- 上記の特性を生かし、従来のスライド型プレゼンテーションソフトウェアでは達成が難しかったプレゼンテーション上のニーズに対応する。

以下項目ごとに説明する。本研究ではプログラムの設計パターンとして広く用いられている Model-View-Controller パターンから着想を得て、プレゼンテーションのストーリーをモデルとして扱い、それをどう画面に表示するべきかのビューと分離することを試みた。これにより、モデルであるストーリーを表示方法の記述と混同することなく検討することが可能となる。また「ストーリーのうち、画面に何を表示するべきか」という点を明確に意識して定義するため、従来のスライド型プレゼンテーションで起こりがちだった「ストーリーの検討をスライドの作成と混同する」「本来ストーリーとして記述すべき“話す内容”をスライドに記述する」といった間違いを防ぐことが期待できる。

2項目目に関して、ビューの記述は Scratch[4] に類似したビジュアルプログラミング言語を用いた。これにより、ビューの記述が直感的かつ容易になる。この方法をとることの欠点として特に Swift で記述した場合に比べ柔軟性が失われる点が挙げられる。しかし前述したようにプログラミング言語を用いてプレゼンテーションの記述を行った経験から通常のプレゼンテーションにおいて必要なプログラミングのパターンが抽出できていたため、それらをサポートすることでビジュアルエイドの記述が可能であると判断した。

3項目目に関し、従来のスライドを利用したプレゼンテーションソフトウェアではスライド単位でしか情報の表示が制御できないため実現が難しい機能があった。例えばスライド内でユーザ操作に応じ逐次的に表示を変更しているのに、プレゼンテーションソフトウェアからの Twitter への投稿がスライド単位でしか行えないなどである。またプレゼンテーション時に「次に何を話すか」を参照する際もノートもしくは次スライドしか参照できず柔軟性に欠けた。そうした問題を解決することを狙った。

2.4 システム設計

開発したアプリケーションのスクリーンショットを図1に示す。

本システムのインタフェースはモデルエディタ、ビューエディタ、プレゼンテーションコントローラの三つの部分に別れている。モデルエディタは、基本的にテキストエディタである。プレゼンテーションのストーリーをこの上で編集する。このエディタには、画像や動画などをドラッグすることで、そのパスを含めることができるが、基本的には自然言語でストーリーを記述する。

ここからプレゼンテーションのビジュアルエイドを作成する際には、モデルエディタ上で画面に表示したい部分を選択し、引用ボタンを押す。すると引用したテキストの前後に括弧が表示され、括弧内のテキスト末尾を拡張子として解釈し、そのタイプ（画像、動画、文章）に対応するプログラムコンポーネントがビューエディタに追加される。画面上での表示位置、順番、フォントサイズなどは全てこのビューエディタ上で設定する。次にビューエディタ上でストーリーから引用されたデータの「見栄え」をビジュアルプログラミングを使用して設定していく。設定された結果は、プレゼンテーションコントローラ部分にリアルタイムに表示されるので、編集結果をただちに確認することができる。

以下実装詳細について述べる。

2.5 実装詳細

2.5.1 ビジュアルプログラミング言語

本システムでは、大別して以下に示す6種類のコンポーネントを用いてビューを記述する。

1. 表示：テキスト/画像/動画の表示
2. Wait: 停止してユーザの操作を待つ
3. 設定：背景色、表示要素を配置する方向、表示要素の削除
4. 再生：動画の再生
5. Tweet: Twitter に投稿する文章
6. ノート：リモコン上に表示する注釈

動作の基本は、画面上に複数の要素を並べて表示しユーザの入力を待つ、というものである。ユーザの入力がなされると、次のコンポーネントで指定された動作を行う。それぞれのコンポーネントでは、表示する際のフォントサイズ、背景色、サイズ、画面上での位置などを GUI を用いて設定することができる。画面上の位置指定は、絶対位置での指定も可能である。しかしこれまでに行ったプレゼンテーションの経験から、多くの場合表示要素を並べる方向を指定した上で、単純に追加し、レイアウトはシステム側で自動的に行うことで十分な場合が多い。一例としての記述を図2に示す。



図 1. 開発したアプリケーションのスクリーンショット



図 2. ビジュアルプログラムの例

以下上記例での動作を説明する．1でそれまでの画面表示要素をクリアする．次に2で指定された画像を背景に表示する．ここまで動作したところで3でユーザのインプットを待つ．ユーザがプレゼンテーションを進める入力をするすると，4で次に表示する文字を見やすくするため背景画像を半透明の面でマスクする．5で文字を表示．ここでフォントのサイズと文字列の左右方向の配置を設定している．6でユーザの入力を待つ．ユーザが入力を行えば，7で別の文字列を表示し，8でユーザの入力を待つ．このように本システムでは直感的に画面上の表示を設定することができる．仮に最初から全ての要素を表示したければ3と6のWaitコンポーネントを削除するだけであり，極めて簡単に一括表示，あるいはユーザ操作に合わせた逐次表示を選択することができる．

またこれらの複数のコンポーネントをモジュールという単位にまとめることができる．この機能により例えば“Introduction”，“Demo”，“Algorithm”などの意味のある単位ごとにプレゼンテーションの記述，管理が可能となる．

Tweet機能を用いることにより，モデル上の任意の記述をハッシュタグをつけた上でTwitter上にタイミングを指定して投稿可能である．またこの機能はプレゼンテーションを一旦終了した後は，プレ

ゼンテーション全体を再読み込みするまで機能しない。これによって、質疑応答の際に情報をブラウザしている際に再度情報が Tweet されることを防ぐことができる。

ノートについては次項で詳細に記述する。

2.5.2 リモコン用インタフェース設計

プレゼンテーション中に画面に表示する内容はプレゼンテーションを視覚的に補助するものという考え方に立ち戻れば、画面に情報を表示するタイミングは喋りと意図的にずらすことが必要な場合がある。例えば「次に示す数字はショッキングかもしれませんが、現実です」と述べた後に数字を画面上に表示するような場合である。ここでは聴衆に深く印象付けたい内容を表示する前に、予告を行なっている。このような場合に、画面表示を進める前に自分がしゃべる内容を確認したくなることが考えられる。既存のプレゼンテーションソフトウェアでも、iPhoneなどをリモコンとして使用した場合、プレゼンターに対してそのような機能を提供するものがある。例えば Apple 社の Keynote では、リモコンアプリケーション上に「現在のスライド」「次のスライド」「現在と次のスライド」「現在のスライドとノート」「次のスライドとノート」を表示するオプションがある。しかしながら、こうした場合にプレゼンターにとって必要なのは「次に何を話すか」を確認する機能である。「次のスライド」を表示する機能は、そうした目的に対して役立つ場合もあるが、そもそもスライドは小さなリモコン上の画面で表示するために作られていないため、プレゼンテーションのプレッシャーの中でその内容を読み取ることが容易であるとは限らない。また次に表示するものが動画である場合、その代表画像だけを見ても自分が何を話すか思い出せない可能性が高い。

そのため、本システムではリモコン上に表示する内容を設定するノートコンポーネントを有している。このコンポーネントを用いることにより、次の表示に進む際に利用する「キュー」の情報をストーリーの中から設定し、それをリモコン上だけに表示することができる。さらにこのノートはプレゼンテーションのプレッシャーの中でもリモコン上で容易に認識できるよう文字サイズなどが大きめに設定されている。リモコンとして使用する iPhone の画面を長押しすることで、次画面が表示される。次の Wait コンポーネントまでの間にノートが定義されている場合には、次画面の代わりに設定されたノートが表示され、プレゼンターは次に話すべき内容を確認することができる。リモコン上の表示例を図 3 に示す。

この機能を用いることで前述したように意図的に喋りのタイミングと画面送りのタイミングをずらすことが可能となる。また前述のパターン 3-c のプレゼンテーションを行う場合にも、記憶が飛ぶことを

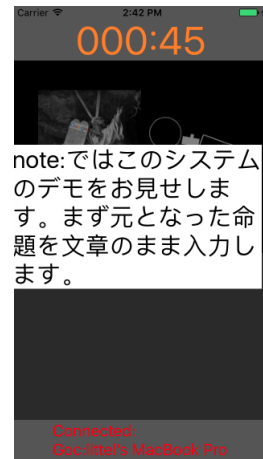


図 3. リモコン上のノート表示例

恐れないうプレゼンテーションの補助として用いることができる。

2.6 実装システム

本システムは Mac 上で動作するアプリケーションとして動作し、Swift で記述されている。リモコン用のアプリケーションは iOS アプリケーションであり同じく Swift で記述されている。Mac 上のアプリケーションとリモコンアプリケーションの接続には Multipeer Connectivity を用いており、Wifi もしくは Bluetooth で接続が可能である。

3 システムの使用例

実装したシステムを用いて社外向けのプレゼンテーション (ARG 第 10 回 Web インテリジェンスとインタラクション研究会) を行った。そこで得られた知見について以下に列挙する。

- ストーリーと、表示仕様を分離するコンセプトは期待通り機能した。
- ビジュアルプログラミングによるビューの記述も可能であり、リアルタイムで画面のプレビューが見られるのはプログラムを行う上で大変便利なことが確認できた。
- プレゼンテーションの本番中何度か「次に何をしゃべるか」とつまった場面があったが、次画面ノート表示機能により支障なくプレゼンテーションを進めることができた。
- 質疑応答の際に、途中の画面を見せようとした場合現在の方式では簡単に指定することができない、という問題がある。

信頼性において市販ソフトウェアより劣る自作システムを外部向けのプレゼンテーションに用いるといった心理的な重圧は別として、上述のように概ね

当初狙った機能は実現できていると考えられる。特にリモコン上に次に見せるべき内容のノートを見ることができる機能は大変便利であるとともに、心理的重圧を軽減してくれる効果が確認できた。今回モジュールを導入したことにより、ある程度長いプレゼンテーションであってもモジュールを用いて構造化し、構成を検討するのが便利になったことが確認できた。しかしそれと裏腹に生じた問題として、全ての表示を一覧する機能の実現が難しくなった。そのため質疑応答の際に任意の画像、テキストなどを瞬時に呼び出すことができず戸惑うという欠点が判明した。ただ同様の問題は従来のスライドを利用したプレゼンテーションソフトウェアにも（程度は異なるが）存在する。この解消方法は今後の課題としたい。また作成したモデルは、文中に使用する画像などへのリンクを含んでおり、少し手を加えればハンドアウトとして活用できることがわかった。モデルに対し、プレゼンテーション用のビューとハンドアウト用のビューなど複数のビューを定義することを可能にすれば、プレゼンテーション用ビジュアルエイドとハンドアウトを同時に作成できる可能性があると考えられる。

4 関連研究

プレゼンテーション用のプログラムに関する研究は多いが、それらのほとんどは従来型のスライドベースのプレゼンテーションソフトウェアを前提としている。また PowerPoint, Keynote などのプレゼンテーションソフトウェアを用いず、テキストベースでプレゼンテーションを記述するシステムも使われるようになってきている。代表的なものとして、Deckset [1] および reveal.js [2] を挙げる。Markdown でプレゼンテーションを記述することが可能である。この際プレートのテキストで表示内容を記述し、表示の仕様（フォントサイズなど）は最小限の記号で指定する。これはある程度プレゼンテーションの画面に表示する「内容」と「見え方」を分離する試みと考えることができコンセプトにおいて本システムと共通する点がある。逆に本研究と比較した場合、ソースコードの変更を行うたびに確認を繰り返す点が手間となる。これは本研究の前段階として実施した Swift でのプレゼンテーション記述と共通する問題である。このような問題を解消する方法として Live Programming が提案されている。Chan らは Web から得たデータをモデルとし、それに対する操作をスプレッドシートの形式でビューとして設定、結果を即座に確認できる画面を備えたシステムを提案している [3]。本研究はこの Live Programming の考え方をプレゼンテーションに取り入れたものと考えられる。

プレゼンターと聴衆との間のインタラクションに着目した研究も数多く存在する [10][7]。本研究はプ

レゼンターがプレゼンテーションを作り上げるプロセスにフォーカスしている点異なる。宮本らによる LaTeX で記述された論文からプレゼンテーション用のスライドを自動生成する試み [8] は、論文をモデル、プレゼンテーション用スライドをビューと捉えれば本研究とコンセプトにおいて共通する点があると考えられる。

5 むすび

プレゼンテーションのストーリーをモデルとして捉え、それをどのように表示するかの「ビュー」と分離したプレゼンテーションのビジュアルエイドシステムを開発した。今後本システムを用いた社外向けのプレゼンテーションをさらに行い、知見を蓄積、システムに反映させることでより柔軟、かつ強力なプレゼンテーション支援システムの開発を行っていききたい。

参考文献

- [1] Deckset. <https://www.decksetapp.com>.
- [2] reveal.js. <http://lab.hakim.se/reveal-js/>.
- [3] K. S.-P. Chang and B. A. Myers. Creating Interactive Web Data Applications with Spreadsheets. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pp. 87–96, New York, NY, USA, 2014. ACM.
- [4] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond. The Scratch Programming Language and Environment. *Trans. Comput. Educ.*, 10(4):16:1–16:15, Nov. 2010.
- [5] T. Urban. Doing a TED Talk: The Full Story. <https://waitbutwhy.com/2016/03/doing-a-ted-talk-the-full-story.html>.
- [6] カーマイン・ガロ. スティーブ・ジョブス 驚異のプレゼン. 日経 BP 社, 2010.
- [7] 栗原 一貴, 望月 俊男, 大浦 弘樹, 椿本 弥生, 西森 年寿, 中原 淳, 山内 祐平, 長尾 確. スライド提示型プレゼンテーション方法論の拡張手法を定量的に評価する研究. 情報処理学会論文誌, 51(2):391–403, Feb. 2010.
- [8] 宮本 雅人, 酒井 浩之, 増山 繁. 論文 LaTeX 原稿からのプレゼンテーションスライド自動生成. 知能と情報: 日本知能情報ファジィ学会誌, 18(5):752–760, Oct. 2006.
- [9] 大坪 五郎. Gozen: プレゼン用「ビジュアル・エイド」のあるべき姿. 第 20 回インタラクティブシステムとソフトウェアに関するワークショップ, 2012.
- [10] 井上 良太, 白松 俊, 大園 忠親, 新谷 虎松. 発表中の資料へのフィードバックに基づくインタラクティブプレゼンテーションシステムの実現. 情報処理学会論文誌, 56(10):2011–2021, Oct. 2015.