T2R2 東京科学大学 リサーチリポジトリ
Science Tokyo Research Repository

論文 / 著書情報
Article / Book Information

題目(和文)	Web プログラミング言語教育のためのプレゼンテーションシステムの開発と有効性の評価
Title(English)	
著者(和文)	上西秀和
Author(English)	Hidekazu Kaminishi
出典(和文)	学位:博士（工学）, 学位授与機関:東京工業大学, 報告番号:甲第9666号, 授与年月日:2014年9月25日, 学位の種別:課程博士, 審査員:室田　真男,中川　正宣,西原　明法,中山　実,西方　敦博
Citation(English)	Degree:, Conferring organization: Tokyo Institute of Technology, Report number:甲第9666号, Conferred date:2014/9/25, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Powered by T2R2 (Science Tokyo Research Repository)

東京工業大学
大学院社会理工学研究科
人間行動システム専攻

博士論文

Webプログラミング言語教育のための
プレゼンテーションシステムの開発と有効性の評価

指導教員　室田　真男 教授

平成26年3月

提出者

氏名　上西　秀和

平成 25 年度　博士論文内容梗概

Web プログラミング言語教育のための
プレゼンテーションシステムの開発と有効性の評価

指導教員　室田　真男　教授
提出者　　上西　秀和

本論文では、講義形式によるプログラミング教育をより効率的に行うためのプレゼンテーションシステム「CodEx」を提案し、その有効性を実証することを目的とした。プレゼンテーションを用いた講義を行うためには、スライド表示・プレゼンテーション実行機能と、作成・編集するための機能が必要である。前者として「CodEx」を、後者として「CodEx GUI Editor」を開発した。

CodEx は、スライド上に表示されたコードボックスに HTML/CSS/JavaScript のサンプルコードを示しながら、実行ボックス上にその実行結果を表示することができるプレゼンテーションソフトウェアである。また、コードボックス上のサンプルコードは、プレゼンテーション中であっても編集することが可能である。

CodEx GUI Editor は、CodEx プレゼンテーションの作成を WYSIWYG に作成するためのプレゼンテーション作成・編集ソフトウェアである。

開発したソフトウェアそれぞれに対してユーザビリティ評価を行った。CodEx の評価としては、受講者・教師それぞれの立場からの評価である。アンケートの結果、要求仕様のほとんどの項目をみたしていることを確認した。Editor の評価として、容易に作成可能であるかどうか、ユーザビリティ評価を行った。結果、全員がスライドを作成できた。一方で、ボックス類のレイアウト変更機能については、インタフェースをより洗練し、使いやすくする必要性がアンケートにより指摘された。

さらに、CodEx を用いた授業の効果を測定するための実験を行った。実験では被験者を 2 群に分け、CodEx を用いた提案手法によるセクションと用いない従来手法によるセクションに分けて授業を交互に行い、それぞれのセクション後に授業内容に関するテストを行った。事後テストと成績と質問紙の回答から、CodEx はプログラミングの講義のうち、メカニズムを教える上で教授効率が高い可能性が示されたが、単純記憶に関する項目では、学習効果に差がみられないものと思われた。

これらの結果から、本論文では・スライド上に表示されたコードボックスにプログラムのサンプルコードを示しながら、実行ボックス上にその実行結果を表示し、提示することができる・教師は、CodEx を用いて、プログラムの動作メカニズムを伝えるためのスライド・プレゼンテーションを作成できる・受講者は、教師の説明を聞きながら、プログラムの動作メカニズムを理解することができる。との結論を得た。

目 次

第 1 章　序論 **5**
1.1　研究の背景 5
1.1.1　高度情報化社会とプログラミング言語教育 5
1.1.2　一般情報処理教育におけるプログラミング言語教育 7
1.1.3　プログラミング言語教育の学習・教育支援 8
1.2　インストラクショナルデザインからみたプログラミング言語教育 . . . 11
1.2.1　I1: 学習者の注意を喚起する 12
1.2.2　I2: 学習者に目標を知らせる 12
1.2.3　I3: 前提条件を思い出させる 12
1.2.4　I4: 新しい事項を提示する 12
1.2.5　I5: 学習の指針を与える 13
1.2.6　I6: 練習の機会をつくる 13
1.2.7　I7: フィードバックを与える 13
1.2.8　I8: 学習の成果を評価する 13
1.2.9　I9: 保持と転移を高める 13
1.3　マルチメディアラーニング理論と分裂注意効果 14
1.3.1　Worked Example 14
1.3.2　分裂注意効果 16
1.4　インストラクショナルデザインと分裂注意効果 18
1.4.1　分裂注意効果の例 19
1.5　教育現場とプレゼンテーションソフトウェア 20
1.5.1　教育現場の事情とプレゼンテーションソフトウェア 20
1.5.2　教育現場でのプレゼンテーションソフトウェアの例 21
1.5.3　プログラミング言語教育とプレゼンテーションソフトウェア . 22
1.6　プレゼンテーションの作成 25
1.6.1　既存のプレゼンテーション作成ソフトウェア 25
1.6.2　分裂注意効果をもたらさないためのプレゼンテーション作成 . 25
1.7　研究の目的 26
1.8　用語の定義 28
1.9　論文の構成 29
1.10　本章のまとめ 31

第 2 章　Web プログラミング言語教育のためのプレゼンテーションソフトウェア「CodEx」の開発　32
2.1 CodEx の概要 32
2.2 HTML プレゼンテーションの利点と欠点 32
2.2.1 HTML プレゼンテーションの利点 32
2.2.2 HTML プレゼンテーションの欠点 34
2.3 CodEx の要求仕様 35
2.3.1 C1: 教師がスライドを表示し、学習者に講義ができる 36
2.3.2 C2: 1 画面の利用 36
2.3.3 C3: スライド上に説明文や視聴覚教材を表示できる 36
2.3.4 C4: コードボックスにサンプルコードを表示できる 36
2.3.5 C5: 実行ボックスの表示 37
2.3.6 C6: コードボックスや実行ボックスは 1 スライド内に複数配置できる 37
2.3.7 C7: 実行ボックスにはラベル名をつけることができる 37
2.3.8 C8: コードボックスには、実行先の実行ボックスのラベル名を設定できる 37
2.3.9 C9: コードボックスには、言語の種類を設定できる 38
2.3.10 C10: 実行ボックスにサンプルコードの実行結果を表示できる 38
2.3.11 C11: コードボックスを逐次編集できる 38
2.3.12 C12: コードボックスの編集結果を、実行ボックスに反映できる 39
2.3.13 C13: 実行ボックスには、複数のサンプルコードの実行結果を重ね合わせたものが表示できる 39
2.3.14 C14: 学習者がスライド資料を手元で使って学習できる 39
2.4 CodEx の構成要素 39
2.4.1 コードボックス 40
2.4.2 実行ボックス 40
2.5 CodEx の機能 40
2.5.1 スライド内に表示される要素など 40
2.5.2 各種コードの実行 42
2.5.3 誤りを含むコードの実行 45
2.6 設計の方針 48
2.7 期待される効果 48
2.8 本章のまとめ 48

第 3 章　CodEx のユーザビリティ評価　49
3.1 実験手順 49
3.2 アンケートの結果 51
3.3 考察 54
3.4 本章のまとめ 56

第 4 章　スライド作成・編集ソフトウェア「CodEx GUI Editor」の開発　57
4.1　開発の目的 57
4.2　要求仕様 57
4.2.1　E1: GUI インタフェースによりスライドが編集できる 59
4.2.2　E2: WYSIWYG なスライド編集画面を表示できる 59
4.2.3　E3: プレゼンテーションのプレビューを表示できる 60
4.2.4　E4: コードボックスをスライド上に配置できる 60
4.2.5　E5: 実行ボックスをスライド上に配置できる 60
4.2.6　E6: 外部リソースをスライド内に表示できる 60
4.2.7　E7: プレゼンテーションファイルを読込・保存できる 60
4.2.8　E8: オフラインであっても利用できる 60
4.3　既存ソフトウェアとの比較 61
4.4　CodEx GUI Editor の機能と実装 62
4.4.1　画面レイアウト 62
4.4.2　Editor 機能の詳細 64
4.5　本章のまとめ 67

第 5 章　「CodEx GUI Editor」のユーザビリティ評価　68
5.1　実験手順 68
5.2　実験結果 71
5.2.1　事前アンケートによる被験者属性の確認 71
5.2.2　スライド作成に関する 7 件法アンケート 71
5.2.3　作成されたスライドの評価 73
5.2.4　自由記述 74
5.2.5　プレゼンテーション作成時間 75
5.3　考察 75
5.3.1　スライド作成に関するアンケート 75
5.3.2　作成されたスライド・プレゼンテーションの評価 76
5.3.3　考察のまとめ 76
5.4　本章のまとめ 77

第 6 章　CodEx を用いた講義の評価　78
6.1　実験目的 78
6.2　実験手順 78
6.3　事前テスト・講義・事後テストの内容 79
6.3.1　事前テストの内容 79
6.3.2　講義内容 83
6.3.3　事後テストの内容 83
6.4　結果 84
6.4.1　被験者 85
6.4.2　事前アンケートと事前テスト 85

6.4.3 事後テストの解答時間 85
6.4.4 事後テストのスコア 88
6.4.5 事後アンケート 91
6.4.6 実験終了時アンケート 93
6.5 考察 94
6.5.1 問題の解答時間 94
6.5.2 分裂集中効果と事後テストの結果 94
6.5.3 事後アンケートの結果 96
6.5.4 事後テストの成績・事後アンケートの結果と教授効率 96
6.5.5 実験終了時アンケート 101
6.5.6 考察のまとめ 101
6.6 本章のまとめ 101

第 7 章 結論 102
7.1 まとめ 102
7.1.1 Web プログラミング言語教育のためのプレゼンテーションソフトウェア「CodEx」の開発 102
7.1.2 「CodEx」を用いた講義の評価 103
7.1.3 スライド作成・編集ソフトウェア「CodEx GUI Editor」の開発 103
7.1.4 「CodEx GUI Editor」のユーザビリティ評価 103
7.1.5 CodEx を用いた講義の評価 103
7.2 結論 104
7.3 今後の課題と展望 106
7.3.1 今後の課題 106
7.3.2 今後の展望 106

謝辞 108

本論文に関する研究発表 109

本論文以外での研究業績 111

参考文献 114

第1章

序論

本章では、本研究の背景について述べ、本研究の目的を明らかにする。また、本論文中で用いた用語を定義する。さらに、本論文の構成についても示す。

本章の流れをまとめた図を、図 1.1 に示す。

1.1　研究の背景

1.1.1　高度情報化社会とプログラミング言語教育

「高度情報化社会」と呼ばれて久しい。国内でも、すでに 1960 年代には「情報化社会」の概念が提唱されているが、その流れは現在においても加速している。医療、工業をはじめ、農林水産業や教育・研究といった分野にまでコンピューターの利用が進展している。

これらの高度情報化社会を支えているのは、プログラマー・システムエンジニア（SE）などのコンピューター技術者である。コンピューターの普及は、これまで人手で行っていた作業を自動化し、省力化に貢献したが、そのコンピューターを造るのは、言うまでもなく人間であり、技術者である。高度情報化社会を支えるために、技術者を育成することは社会にとって大きな意義をもつ。

さて、コンピューターを構成する要素は、大きく「ハードウェア」と「ソフトウェア」に分けられる。このうち、「ソフトウェア」を作成するためには「プログラミング言語」が欠かせない。IT 用語辞典 [47] の定義によれば、「ソフトウェアの設計図に当たるソースコードを記述するための言語」となっている。これは、人間がコンピューターを動かすための設計図を、定められた言葉（言語）を使って書くことを意味する。普段、われわれが使っている日本語・英語・ドイツ語・中国語…などの自然言語のように、コンピューターを動かすための言葉が存在するのである。ただし、プログラミング言語は、人間が人間の都合で作った「人工言語」である。

自然言語に多くの種類があるように、プログラミング言語にも、目的に応じて多数の種類（例として、Fortran、C、C++、Java、JavaScript、Perl、PHP など）がある。また、広義にコンピューターを「動かす」ことを考えるならば、コンピュー

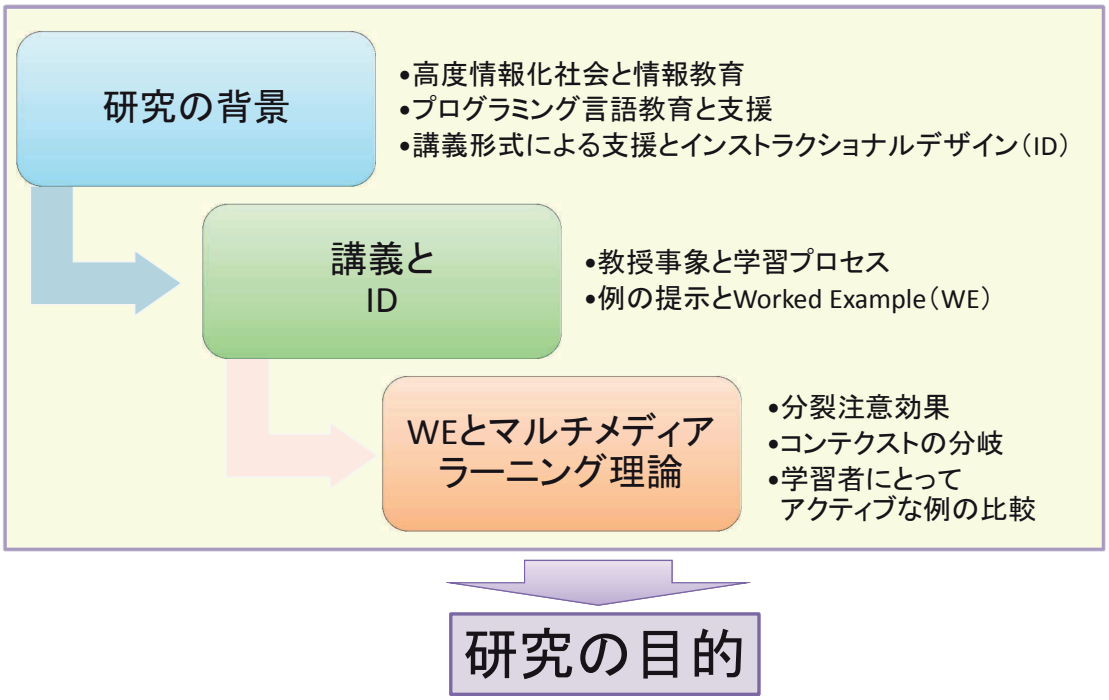
研究の背景
•高度情報化社会と情報教育
•プログラミング言語教育と支援
•講義形式による支援とインストラクショナルデザイン（ID）
講義と
ID
•教授事象と学習プロセス
•例の提示とWorked Example（WE）
WEとマルチメディア
ラーニング理論
•分裂注意効果
•コンテクストの分岐
•学習者にとって
アクティブな例の比較
研究の目的

図 1.1　第 1 章の流れ

ター画面を表示させるために必要な言語（例として、HTML、CSS、SVG）もプログラミング言語に含めるべきであると筆者は考えている。とくに、近年の HTML5・CSS3 といった技術は、かつては JavaScript などを用いなければ表現できなかった動作を記述することができる。そういった意味で、このような拡張的な定義は妥当ではないかと考えられる。

「ソフトウェア」を産み出すコンピューター技術者を養成する上で、プログラミング言語を習得させることはきわめて大きな意義をもつ。高度情報化社会が進むにつれて、教育機関にて情報学を専門に学習した（**情報専門教育**を受けた）人材だけでは情報化社会を支えられなくなってしまった。別の言葉で言うならば、人材が不足しはじめたといえる。筧の推測 [63] によれば、大学における情報専門学科の卒業生は毎年高々12000 人であるが、国勢調査における IT 技術者は 849000 人であるという。

このように、数値の上からも情報専門教育を受けた人材だけでは、高度情報化社会を支えられていないという現実がある。そのため、情報専門教育を受けた者だけでなく、より多くの人々に対して、教育によってプログラミング習得を促すようなしくみが必要となった。特に、高等学校や大学学部教育などで、コンピュータやプログラミングについて学ぶ機会を設ける（**一般情報処理教育**を学ぶ機会をつくる）ことが、高度情報化社会に貢献する人材を輩出するうえで重要であると筆者は考える。

1.1.2　一般情報処理教育におけるプログラミング言語教育

情報処理教育における知識体系（BOK：Body of Knowledge）が、国内外においてまとめられている。これらは、情報処理教育に対して、モデルとなるカリキュラムを提案・提供する。

例として、ACM と IEEE-CS の標準カリキュラム BOK である Computing Curricula 2005（CC2005）は、コンピュータ系学部におけるカリキュラム BOK の共通部分やそれぞれの違いについてまとめそれぞれのカリキュラムによる成果の特徴について述べている [20]。これに対して、各カリキュラムについて詳細に述べた BOK として Computer Science Curriculum (CS) 2008[36]、Computer Engineering (CE) 2004[37]、Software Engineering (SE) 2004[38]、Information Technology (IT) 2008[15]、Information Systems (IS) 2010[21] がある。このようなカリキュラム BOK では、あくまで情報専門教育について述べられており、一般情報処理教育に言及されていない。

日本では、情報処理学会が、上記の CS、CE、SE、IS、IT に対応し、整合性を保ちながら「情報専門学科におけるカリキュラム標準 J07」という BOK を策定している [59]。

その一方で、先に述べたように、情報専門学科で学ぶ学生数と比較して、一般情報処理教育を受講する対象となる学生数が圧倒的に多いという現実がある。そのため、情報処理学会では、情報専門学科用の BOK に加えて、一般情報処理教育カリキュラムを普及・啓蒙するための機会として一般情報処理教育（General Education, GE）カリキュラムを策定（GEBOK）した [51][50]。

GEBOK では、『将来，高度情報社会において中核となる大学生に対して、情報およびコンピュータに関する基礎理論や概念および応用知識を理解させるとともに、それらを自由自在に活用できる能力を身につけさせることとする。』とうたわれている。このことを踏まえ、GEBOK では

- 科目ガイダンス（当該大学のネットワーク環境と情報倫理規定）
- 情報とコミュニケーション
- 情報のディジタル化
- コンピューティングの要素と構成
- アルゴリズムとプログラミング
- データモデリングと操作
- 情報ネットワーク
- 情報システム
- 情報倫理とセキュリティ
- コンピュータリテラシ補講

を具体的内容として取り上げている。

このカリキュラムのうち、「アルゴリズムとプログラミング」では、

『アルゴリズムとは何かについて説明でき、簡単な問題に対して、アルゴリズムを考案できるようになることを目的とする。また、変数、制御構造等のプログラムの構成要素を理解し、繰返しを含む簡単なプログラムの作成ができること。また，その体験を通してプログラミングの難しさを理解することを目的とする。このため、コア時間には、体験のための演習時間数を含む。』

とあるように、学生がプログラミング言語について学ぶ必要性が示されている。また、「情報ネットワーク」では、WWW の仕組みについて学ぶこととされており、その際に学生が HTML や CSS の仕組みについて学ぶ機会がある。

1.1.3　プログラミング言語教育の学習・教育支援

他の教育支援と同様に、プログラミング言語学習においても、「講義時」「授業内」における学習支援と「自学自習」における学習支援とに分類できる。プログラミング言語学習における授業では、教師による説明が簡略で、授業時間の大半を演習に充てている例も多い。そこで、特に講義による説明を支援するような場合を「講義時」、授業内の演習においての学習支援を行っている場合を「授業内」と分類した。

自学自習支援

プログラミング言語の学習においては、自主的な学習が重視される傾向が強い。教育支援においても、自学自習支援を行う試みが多数報告されている。

Web 上で利用できるプログラムの実行環境を準備し、学生が自学自習したり、宿題の提出等を管理したりすることができるシステムが提案・開発されている。例として、森田 [61] は、「データ構造とアルゴリズム論」などの授業において、自動採点機能を有する Web テストを導入した。この Web テストは、学生に繰り返し受験させるものである。受験ごとに各設問の正答率などの学習履歴を学生に提示し、学生は自分の理解度を相対的に把握できるようになっている。同様の試みに篠原・宮武 [55] があるが、こちらでは教員がレポートを条件検索する機能や、コメントや評価点を書き込み学生に提示する機能を有している。このような試みにより、学生が反復学習を行う機会を提供するとともに、教師の負荷軽減を図っている。

学習者の解答の正誤判定に着目した実践として、松本ほか [56] の研究がある。ここでは、テストケースを教師があらかじめ作成しておき、学習者が送信したプログラム全体をテストする。学習者には、テストの判定結果やテストの目的、ヒントがフィードバックとして与えられる。提案された手法では、正しい解答を誤ったものとしてフィードバックしてしまう可能性があるが、80 %は正しいフィードバックが返されたという。その一方で、一定の学習者には、システムによってプログラムのバグを発見することができるという効果が確かめられている。

プログラミング言語の自主学習を促進する試みは、システムに頼った学習支援に限らない。教師がいなくても自学自習できるようにする試みとして、動画教材が多数開発されつつある。国内ではドットインストール（dotinstall.com）[9]、海外では thenewboston [3] などの試みが知られている。これらは、学習内容を数分程度の短編のビデオとして提供することで、学習者が好きな時間に集中して観ることができるように工夫している。また、このようなビデオ教材は、授業開始前の予習用に利用することが考えられる。ビデオ教材を授業の予習用に公開した例として、Carlisle の実践 [4] がある。この研究では、Java 入門編の授業の予習教材として短い 21 のビデオ教材を制作し、学生に予習するように指示した。その結果、学生は、ビデオ教材が学習に役立つとアンケートで回答した。教員視点からでは、予習を行う学生が（ビデオ教材による予習だけでなく、教材を読む予習も）増えた。また、予習を行った学生とそうでない学生間には、人数が少ないために統計的有意差は出なかったものの、成績の差も見られた。

授業内支援

プログラミングの授業においては、教師がすべての学生の面倒を見ることが、時間的・人員的制約から不可能である場合も多い。そのため、教師の負担を軽減する「授業内支援」の意義は大きいと考えられる。

西谷ほか [62] は、授業内の演習において、教師が学生の誤答に対してフィードバックを行うシステムを開発している。このシステムでは、同じ誤答をした学習者を一

つのグループにまとめ、教師のアドバイスを同時にこれらの学生に伝達する。この仕組みにより、教師によるアドバイスを「高い頻度でかつ効率的に」学生に送信することができるなど、教師の負担軽減と集合教育の限界を改善することに寄与している。

高山 [53][54] は、Web 環境を利用したプログラミング教育支援システムを開発しているが、この試みでは JavaScript のライブラリ processing.js を用いることで、グラフィカルな出力を行うプログラミング環境としている。Web 環境を利用したプログラミング教育支援は、先に述べた「自学自習」型の学習支援に用いられることも多いが、授業内の演習において用いられる例も多い。

ここまでに取り上げた例は教師の負荷を軽減することに重きが置かれている一方、教師の負担が多少増えても学生により深い理解を与えるべきであるとの主張もある。そのような試みの例として、新開・宮地 [60] は、C 言語のプログラミングにおけるプロセスの重視と評価活動を取り入れた教育の効果について報告している。この報告では、理解度を把握するための質問カードの活用や、評価活動を取り入れた課題演習を試みている。その結果、アルゴリズム作成までのプロセスを重視した指導がアルゴリズムを段階的に考える力を向上させること、学生同士の他者評価活動は、アルゴリズムを改善する力やプログラムを読む力を育成するのに役立っていることなどが明らかになっている。

講義時の支援

このように、プログラミング言語教育における学習支援には、さまざまなものが提案されているが、一斉講義による教育場面を想定した学習支援については、ほとんど提案されていない。講義形式による教育では、受講者が受身になってしまう弊害があるものの、同時に多数に向けて知識を伝達できる効果は大きいといえる。また、次節で述べるように、インストラクショナルデザインの観点から見た場合にも、教授行動が適切であるような外的事象が学習者の学習プロセスを支援する。講義形式により、大人数に対して学習プロセスを支援することができると考えられる。

講義形式の授業を行う場合には、黒板や OHP、コンピュータを介したプレゼンテーションソフトウェアがひろく利用されている。プログラミング言語を教育・学習する場合には、コンピュータの利用は必須ではないが、現代において、プログラミング言語を記述し、実際に動作させるのは、ほぼすべてコンピュータを介して行われる実態がある。このことから、講義においてもコンピュータを利用できることがのぞましく、また、当然のように考えられているため、本論文では、コンピュータとプレゼンテーションソフトウェアを用いた教育について検討する。

以下では、インストラクショナルデザインの観点から、講義形式の教育について議論する。次に、マルチメディアラーニング理論から、プログラミング言語教育をプレゼンテーションソフトウェアを用いて行うための要件について検討する。

1.2　インストラクショナルデザインからみたプログラミング言語教育

「講義によるプログラミング言語教育」をより明確に定義づけするために、本論文ではインストラクショナルデザインの観点を用いることを考える。講義は、学生に対して外的事象を中心として学習を支援するものであるから、「9 つの外的教授事象」を適用し、検討してみよう。

インストラクショナルデザイン論において、9 つの外的教授事象が明らかになっている [48][13]。Gagne や Briggs は、インストラクションをこれらの外的事象を中心にして構築することで、学習を達成するための内的事象の働きを促すようにすることを提案した。彼らによれば、これらの外的事象を、レッスンや学習活動を構築するための枠組みであるとみなしている。よい学習者にとっては、必ずしもインストラクショナルデザイナーが常にすべての外的事象を提供する必要はないが、これらの外的事象の 1 つ 1 つがどのような形でもたらされたとしても、すべての学習者の学習プロセスを支援することは明らかであるという。

この観点に立てば、講義を通じて学習者の学習プロセスを支援することができると考えられる。ここで、これら 9 つの外的事象が、プログラミング言語教育において具体的にどのような事象にあたるかについて述べる。

インストラクションとは、学習の内的処理を支援するように設計された学習者の外側に存在する事象の集合体である。教授事象は、学習者の情報処理プロセスを稼働させるためか、あるいは少なくとも情報処理プロセスの発生と並行して提供され、その処理の支援を行うために設計されるものである [48]。

教授事象と、それぞれの学習プロセスの保持は、以下のようになる [48][13]

学習者の注意を喚起する　：神経インパルスのパターンの受容

学習者に目標を知らせる　：実行制御プロセスの活性化

前提条件を思い出させる　：以前に学習したことを作業記憶に想起

新しい事項を提示する　：選択的知覚ができるように特徴を強調

学習の指針を与える　：意味的符号化、検索のためのヒント

練習の機会をつくる　：反応の組織化の活性化

フィードバックを与える　：強化の確保

学習の成果を評価する　：検索の活性化、強化を可能に

保持と転移を高める　：検索のためのヒントと方略を提示

ここで、すべての教授事象がこの順序どおりに学習者に提示されるとは限らないことに注意する必要がある。また、あくまで教授事象の役割は、内的な情報処理を刺激することであり、内的な情報処理そのものにとって代わるものではない。

この教授モデルをプログラミング言語教育における具体例としてまとめたものを、表 1.1 に示す。

1.2.1　I1: 学習者の注意を喚起する

学習者の注意をひきつけるために、教師が受講者の興味・関心する事柄に関連した項目を確認・発問する。あるいは、受講者にとって興味深いサンプルコードを提示するなどの具体例が考えられる。

1.2.2　I2: 学習者に目標を知らせる

受講者が学習を進める上で、教師が学習目標を伝えることは重要である。すなわち、学習者に達成してほしい知識やスキルが何であるかを学習者に伝えなければならない。プログラミング言語教育においては、適切な課題を与えることで、目標を与えることになるだろう。

1.2.3　I3: 前提条件を思い出させる

受講者に既知の知識を思い出させる手法としては、過去に学習したサンプルコードや、その実行結果を提示することが考えられる。ただし、インストラクショナルデザインの上では、以前に学習したことが学習事象の一部として自由に活用できる状態に置かれている必要があることを重視する。そのため、教師がただ情報を提示するだけでなく、教師が再認形式の発問をしたり、あるいは小テストをするなどして過去の知識を利用し、「活用」できるような形で提示するべきである。

1.2.4　I4: 新しい事項を提示する

適切な刺激を受講者に与える。プログラミング言語教育においては、新しい構文・文法の提示や、サンプルコードの提示、その実行結果の提示が考えられる。

ここで、新しい事項の与え方には、2 通りの方法がある。すなわち、

- ルールを示してから事例を提示する
- 事例を先に示してからあとでルールを提示する

の 2 通りがある [48]。これをプログラミング言語教育の例に当てはめれば、「新しい構文・文法を提示し、サンプルコードを提示する」のが前者、「サンプルコードを提示し、そのコードを用いて新しい構文・文法について習う」のは後者に当たる。

1.2.5　I5: 学習の指針を与える

新しい事項を提示することに対して、この段階では、学習者は学ぶための文脈をつくり上げることとなる。前段階で与えた「定義された概念」に対し、分類のためのルールを応用することによって「分類」できるようにする必要がある。

プログラミング言語教育の場合には、前項で学習した内容に類似した構文・文法を再提示したり、類似した実行結果を再提示することによってこれらが可能となると考えられる。あるいは、前項のサンプルコードを臨機応変に変更し、提示することで類似した例を提示することができる。

1.2.6　I6: 練習の機会をつくる

プログラミング言語学習においては、前項と同様に類似した例を提示することが考えられる。ここで、前項との違いは、学習者にどのように問題を解けばよいのかわかっていることを実際に見せてもらう。そのために、学習者が考え、解答することができる形で発問する。

1.2.7　I7: フィードバックを与える

プログラミング言語教育の場合、実行環境にコードを書いて実行すれば、その出力結果を知ることは（コードの実行時間が十分短い場合には）容易である。一方で、コードの出力結果だけ正しいが、途中の過程が正しくない場合も起こりうる。このような事例を防ぐために、間違った方向性を指摘する教師の働きかけが重要である。単に正誤を伝えるだけでなく、ときには矯正的なフィードバックが必要になることもある。

1.2.8　I8: 学習の成果を評価する

学習の成果を評価する上で、受講者に成果物を作成させる。具体的には、プログラミング演習を行わせるとよいだろう。さらに、講義形式においては、学習者が解答した内容について、教室内で発問を行い、それをクラス内で共有することもできる。

1.2.9　I9: 保持と転移を高める

学習した内容（ここでは、知識またはスキル）を忘れないようにするため、あるいは学習した内容を必要なときに呼び出す能力を受講者の中で拡張するために、学習したときと同じ文脈で情報または知識を思い出させるような練習などを受講者に行わせる。講義においては、教室内の学習者を指名し、類題を学習者に解いてもらうように指示する。このような方法をとることで、ある学習者が解答するプロセス

を、他の学生が教室内で共有しながら学習した内容の保持と転移を高め、学習のメタ化を図ることができる。

なお、プログラミング言語教育の講義に焦点を当てた本論文の範囲を超えるが、プログラミング言語教育の場合には、最終的により実践的な演習問題を通じて、プログラミングスキルが身に付くようにする必要がある。

表 1.1　プログラミング言語教育における教授事象と学習プロセスの関係

	事象	プログラミング言語教育における具体例
I1	学習者の注意を喚起する	注意喚起のための発問、 興味深いサンプルコードの提示など
I2	学習者に目標を知らせる	課題の提示など
I3	前提条件を思い出させる	再認方式の発問、 過去のサンプルの提示
I4	新しい事項を提示する	文法の提示/実行結果の提示/サンプルコードの提示（ルール→事例/事例→ルール）
I5	学習の指針を与える	文法の再提示/実行結果の提示/ 他のサンプルコードの提示
I6	練習の機会をつくる	提示した問題の類題の提示
I7	フィードバックを与える	類題の解答を提示
I8	学習の成果を評価する	学習者が解答した内容についての発問と その結果の共有
I9	保持と転移を高める	類題の提示と発問

1.3　マルチメディアラーニング理論と分裂注意効果

先に、教師側からみたプログラミング言語講義について、インストラクショナルデザインの概念を用いて説明した。教師のこれらの活動は学習者の学習を支援するためにある。さらに、マルチメディアラーニング理論 [14][5] に従い、プログラミング言語教育におけるスライド資料の提示方法について検討する。

1.3.1　Worked Example

Worked Example とは、段階を踏んだ演示であり、作業がどのように行われるかや、問題がどのように解決されるかといったものを説明するのに使われる。Worked Example は学習者の手続き的なスキルを構築するために使われる。例えば表計算ソフトウェアや、ある交渉をどのように進めるかといった戦略などを教授する際に使われる。[5]Worked Example については、代数学の問題解決についての研究を皮切りに、戦略的なスキルを身につける必要のある多くの分野で（例としてスタイルデザイン、電気分野のトラブルシューティング、幾何学、教育原則の応用など）、効果的であることがわかっている。

Clark・Mayer[5] は、これらを以下の原則にまとめている。以下の説明では、プログラミング言語を講義で教える場合についての例にも触れる。

W1: Worked Example から練習問題に移行せよ 最初は完成に近い状態の例を与え、徐々に学習者が自ら問題を解決していくようにする。

W2: 例を学習者が自己説明できるように促す Worked Example により想定される問題点に、学習者が学習内容を深く考えることなく振り返らないことが挙げられる。それを防ぐために、学習者に、Worked Example を与えたら、その例を学習者自身で説明させるように促す必要がある。例えば、教師が与えた Worked Example に対して学習者が説明させるような問題を加えたり、積極的な観察を通して学習者自身で説明できるように奨励するなどである。講義中に教師が受講者に積極的な発問を行うなどすることが考えられる。

W3: 状況に応じて Worked Example に説明指示文を含ませるようにせよ 状況に応じて、Worked Example に指導的な説明文を含むようにする。スライド上の説明文や、教師の口頭での説明が考えられる。

W4: Worked Example にはマルチメディア原則を適用せよ Worked Example は、マルチメディアラーニング理論に従うべきである。

W4-1: 関連のある視覚教材を利用せよ テキストだけでなく、視覚教材を利用すべきである。Clark・Mayer は、例として、表計算ソフトウェアの式を入力することを教える際には、データとともに表計算ソフトウェアのスクリーンショットを提供することが、関連する視覚教材の例であると挙げている [5]。プログラミング言語教育でも同様に、スクリーンショットを提供することができるが、状況が許せば、実際の動作例を提示することもできる。

W4-2: 視覚教材は音声か文章、どちらか片方で説明せよ 音声と文章を同じものにして同時に提供すべきでない。

W4-3: 視覚教材と統合された説明文を提示せよ 後に述べる分裂注意効果を防ぐために、視覚教材と説明文は統合的に示すべきである。これを実現するには、スクリーン内に表示する情報について、自由度の高い配置が行える必要がある。

W4-4: 内在する原理に着目したチャンクに分解せよ 順番を追って情報を与えるべきである。教師が順番を追いながら受講者に説明する必要がある。

W4-5: 学習者の学習ペースに合った複雑な例を提示せよ 一斉講義では難しい問題であるが、学習者の学習ペースに合わせて、例を複雑にしていく必要がある。一斉講義では、教師が学習者の様子を観察し、学習者のペースを把握するとともに、段階を追った例・問題を提示しなければならない。

W4-6: 学習者が不慣れな内容では事前練習を提供せよ 学習者が不慣れな内容では事前練習を提供するか、学習者が慣れた内容を利用せよ。学習の導入部分において、学習者が不慣れな内容については事前練習を行わせる工夫が必要である。

W5: 学習の広範囲な転移をサポートせよ インストラクショナルデザインでも触れられたように、学習の転移が重要であるが、特に、様々なコンテクストにおける例を示すことが重要である。

W5-1: コンテクストが様々に分岐する Worked Example を提供せよ それぞれが同じガイドラインを持った 2 つかそれ以上の、コンテクストが様々に分岐する Worked Example を提示すべきである。教師が複数のサンプルコードを示すことで、このような状況を産み出すことができる。

W5-2: 様々なコンテクストでの例を比較できるようなインタラクション 学習者がアクティブに様々なコンテクストでの例を比較できるようなインタラクションを提供せよ。積極的に質問をし、アクティブに様々なコンテクストでの例を比較できるようにすべきである。教師がサンプルコードを編集し、様々な例をアクティブに示しながら、学習者に発問すべきである。

これらの原則をプログラミング言語の講義に当てはめた場合、多くは教師の行動に依存する。ただし、プレゼンテーションソフトウェア側の機能で対応しなければならない項目もある。例として、「W3: 状況に応じて Worked Example に説明指示文を含ませるようにせよ」「W4-3: 視覚教材と統合された説明文を提示せよ」といった機能は、ソフトウェア側の機能で実現するべきである。また、学習の広範囲な転移をサポートするために、コンテクストが様々に分岐することを実現すべきであるが、既存の多くのプレゼンテーションソフトウェアでは、先に述べる教育向けプレゼンテーションソフトウェアでは、様々な解決法が試みられているが、プログラミング言語の講義に特化したものは少ない。

本研究では、講義を行いながらであってもスライド中のサンプルコードを編集可能とし、コンテクストの分岐を可能とする。

1.3.2 分裂注意効果

講義において、説明・サンプルコード・サンプルコードの実行結果といった複数の種類の情報を示す場合、マルチメディアラーニング理論 [14][10] において提唱されている「分裂注意効果」(the split-attention effect、訳語は安藤・植野 [49] を参考とした) について考慮する必要がある。

講義内における分裂注意（split-attention）は、学習者が、物理的あるいは時間的に分断された情報を頭の中で統合しようとしたときに、それらの情報源の間で注意が分断される際に起こる。同じ情報による学習を考えた際、統合された情報を与え

られた学習者のほうが、統合されていない情報を与えられた学習者と比較して高いパフォーマンスを示すとされる。情報が統合されていない場合、与えられた情報を頭の中で統合するのに多くの認知負荷（cognitive load）がかかる。このような外的認知負荷が、分断された情報を統合する際に負の効果を与えると考えられる [39]。Ayres・Sweller[39] によれば、頭の中で統合して理解しなければならないような分断された情報源を学習者に示すとき、情報は統合された状態で示すべきである。

マルチメディア学習における分裂注意効果に関する研究がいくつか行われている。Tarmizi・Sweller[2] は、幾何学の解答の習得について研究した。実験者は、幾何学問題に対して、被験者に統合されたフォーマットによる図形問題と統合されていない図形問題をそれぞれ与えた場合の、問題の習得時間と解答時間を測定し、比較している。また、事後テストのスコアも比較している。この研究によれば、統合されたフォーマットで与えた場合に習得に要する時間が減ることから、統合されたフォーマットは認知不可を減少させると結論づけている。このように、この研究は図的情報と文字情報という、異なる属性をもった情報間における分裂情報効果について調べている。

Sweller や Chandler[30][40] は、コンピュータ上における学習における分裂注意効果の例について調べた。彼らの研究では、コンピュータソフトウェアの利用法を教える際、完全に紙の上で統合された情報を与えたほうが、紙とコンピュータを併用した場合に勝るという結果が得られた。この結果は、コンピュータソフトウェアの使い方の学習にコンピュータを用いることが必ずしも効果的でないことを示している。その理由として、彼らは分裂情報による認知負荷の増大を挙げた。

続いて、Cerpa ら [35] は、先の結果は分裂情報によるよりも、むしろ情報伝達媒体の違いによって起こりうると理由づけた。彼らは、表計算ソフトウェアの使い方について学ぶ際、コンピュータ上に統合された情報による説明書と、コンピュータ画面とともに従来の説明書を併用する場合を比較した。結果として、前者のほうが後者に比べて分裂注意効果を軽減したことを示した。特に、学習後の事後テストでは、コンピュータ上に統合された説明書を与えたグループでは、例えば公式を作成するといったような高度にインタラクションの発生する問題において、有意に高い得点を示した。この研究は、分裂注意効果を防ぐように設計された教材であれば、コンピュータ上であっても効果的に利用できることを示している。これらの研究は、媒体間の違いによる分裂注意効果を比較したものである。

複数の統合して理解しなければならない情報源が、物理的分断だけでなく、時間的に分断されている場合にも、認知負荷が高まることが推測される [39]。例えば、Mayer・Sims[11] は、問題解決作業において、プレゼンテーションとナレーションを同時に行った場合には、アニメーションをナレーションの前に提示したり、ナレーションをアニメーションの前に提示した場合と比較して効果的であったことを示している。Mayer[10] は、文章と図は時間的に分断すべきでないと主張している。

1.4　インストラクショナルデザインと分裂注意効果

先に、インストラクショナルデザインの観点からプログラミング言語講義について、「9 つの外的教授事象」に対する具体例を挙げた。この各項目のうち、分裂注意効果を軽減することに注意しながら講義を行うための条件について考える。

9 つの具体例のうち、分裂注意効果の影響を考える必要があるのは、I4. 文法の提示/実行結果の提示/サンプルコードの提示（ルール→事例/事例→ルール）　I5. 文法の再提示/実行結果の提示/他のサンプルコードの提示　I7. 文法の再提示/実行結果の提示/他のサンプルコードの提示　I9. 類題の提示と発問のように、複数の情報を提示してそれらの統合により学習を進める場面である。これらの場合における条件を図 1.2 に示す。

以下、それぞれの場合について説明する。

I4. 文法の提示/実行結果の提示/サンプルコードの提示　プログラミング言語教育のためには、教師は新しい事項を提示する際に、文法や実行結果、サンプルコードといった複数種類の情報を提示する必要がある。学習者はこれらの情報を統合することで学習を進めることとなる。情報の統合の際に、各情報が分裂的に示されないようにすることが条件となる。

I5. 文法の再提示/実行結果の提示/サンプルコードの提示　前項と同様に、情報の統合の際に、各情報が分裂的に示されないようにすることが条件となるほか、学習の指針情報が問題解決を行うための情報と分裂して示されないようにすることが条件となる。

I7. 類題の解答を提示　従来の問題と比較のために、必要に応じて従来の問題の情報が分裂されずに表示できることが条件となる。類題の解答がなぜそのようになるのかを理解するためには、従来の問題との比較が不可欠だからである。

I9. 類題の提示と発問　前項と同様、必要に応じて従来の問題の情報が分裂されずに表示できることが条件となる。従来の問題と比較することで、より効率的に学習のメタ化を促すことができる。ただし、学習者に対して過剰に情報を与えた場合、学習者が思考停止に陥り、学習が促されなくなるという危険性もある。

表 1.2　プログラミング言語講義における学習の具体例と分裂注意効果軽減のための条件

	学習の具体例	分裂注意効果軽減のための条件
I1	注意喚起のための発問、興味深いサンプルコードの提示など	-
I2	課題の提示など	-
I3	再認方式の発問、過去のサンプルの提示	-
I4	文法の提示/実行結果の提示/サンプルコードの提示（ルール→事例/事例→ルール）	各情報が分裂的に示されないようにする
I5	文法の再提示/実行結果の提示/他のサンプルコードの提示	学習の指針情報が問題解決を行うための情報と分裂して示されないようにする
I6	提示した問題の類題の提示	-
I7	類題の解答を提示	必要に応じて従来の問題の情報が分裂されずに表示できる
I8	学習者が解答した内容についての発問とその結果の共有	-
I9	類題の提示と発問	必要に応じて従来の問題の情報が分裂されずに表示できる

1.4.1　分裂注意効果の例

本研究の目的のひとつとして、プレゼンテーションソフトウェアを用いたプログラミング言語講義において、分裂注意効果を軽減することが挙げられる。前節で述べたように、プログラミング言語を講義する際には、

a ：適切な視聴覚教材・説明文

b ：サンプルコード

c ：実行結果の情報を提示すること

が必要である。そして、多くの場面で、これらを同時に提示するためには、プレゼンテーションソフトウェアや実行環境を教師が切り替える必要がある。すなわち、時間的分断による分裂注意効果の影響を受ける可能性がある。

これらを同時に表示するための解決策の 1 つに、複数のスクリーンを利用することが考えられる。このような複数スクリーンを用いたプレゼンテーションの提示に関する研究が、いくつか発表されている。

例として、Lanir らによる MultiPresenter[29][28] がある。これは、複数のスクリーンや、超高解像度のスクリーンを用いて複数のスライドを同時に表示することがで

きるようにしたものである。ただし、このシステムはプログラミング言語教育のために利用されることを想定していない。

Chang ら [46] は、2 つのスクリーンを用いてプレゼンテーションスライドと開発・実行環境を同時に表示し、分裂注意効果の軽減を図った。彼らは被験者を 2 群に分け、片方のグループは 1 つのスクリーンで授業を行い、もう片方のグループには 2 画面で授業を行った。その結果、2 画面を用いたグループのほうが事前の知識確認テストと事後テストの成績をもとにした比較で有意に高い点数を示した。

Endo・Murota[42] は ZUI（Zooming User Interface）を用いた 2 画面によるプレゼンテーションソフトウェアを開発した。これは、教師が 2 画面を用いてプレゼンテーションスライドと開発・実行環境を表示することを支援するものである。

しかしながら、2 画面を用いて授業を行うことは、物理的制約から難しい場合が多い。講義室に、1 面のみのスクリーンが設置されている場合が多いこと、2 面のスクリーンが設置されている場合でも、2 つのスクリーンに別の映像を表示できない場合があることが考えられるからである。

1.5　教育現場とプレゼンテーションソフトウェア

1.5.1　教育現場の事情とプレゼンテーションソフトウェア

以下、前節で触れたインストラクショナルデザインにおける 9 教授事象を踏まえながら、プレゼンテーションソフトウェアを用いて講義を行うことについて考える。

教育現場でプレゼンテーションソフトウェアが導入されて久しい。

プレゼンテーションソフトウェアは、発表者が画面（モニタや、ほとんどの場合は大衆が見ることのできる大型スクリーン）に文字や図表などを表示し、聴衆に対して話したい内容の伝達を補助するためのツールである。代表的なプレゼンテーションソフトウェアとして、Microsoft 社の PowerPoint が知られており、2004 年時点で一時間あたり 125 万回利用されているとされる [31]。プレゼンテーションソフトウェアは、様々な場面で用いられる。例として、企業が顧客に対して行うプレゼンテーション、学会で発表者が聴衆に対して行うプレゼンテーションなどがある。このような場合には、発表者が一方的に話した後で、聴衆が質問を受けるという形式をとる場合が多い。すなわち、既存のプレゼンテーションソフトウェアは話者が聴衆に対して一方的に説明を行うような場合に重点が置かれている。一方で、これを教育現場で利用する場合には、このような想定と異なる利用ができることが必要とされる。特に、「W5 : 学習の広範囲な転移をサポート」するために、「様々なコンテクストでの例」を提示できることが必要である。

- 教師と受講者間に対話が発生する
- 受講者の理解に応じて、臨機応変に説明を加える場面が多く発生する

このような要求に対して、いくつかの解決策が図られている。例えば、PowerPointでは、「インク」機能を用いて、スライド上に文字を書き込む機能が用意されている。これとタブレット PC を併用することにより、講演者と聴衆の対話をリアルタイムに記録・表示することができる。ただし、教育現場で起こりうる様々は利用ケースに PowerPoint に搭載された標準機能だけでは不十分であるとも考えられ、実際に、教育現場のさまざまな要求に対応したプレゼンテーションソフトウェアが多数開発されている。

1.5.2　教育現場でのプレゼンテーションソフトウェアの例

教育向けプレゼンテーションソフトウェアの例をいくつか挙げる。

まず、タブレット PC を利用して OHP のように追記したり図形を挿入したりでき、臨機応変に説明を加えることが可能なもの [41] がある。これを用いると、学習者に、インタラクティブに学習内容のフィードバックを与えることができるほか、学習者が全体に対して自身の解答を示すような「練習の機会」を与えることができる。

Palette では、スライドの内容が簡単にわかるよう、カードを用いたインタフェースを利用し、必要な説明を必要なタイミングで行えるよう、インタラクティブに授業を行うことができる [33]。例えば、以前に学習した内容のカードを利用することで以前に学習した内容を思い出させ、「前提条件を思い出させる」ことが可能である。また、複数のスライドの中から必要なものを任意の順番で提示できることから、学習者の興味・関心を教師が講義中の発問によって測りながら「学習者の注意を喚起する」内容を提示することも容易となる。

直観的な操作が行え、発表中に編集する機能を強化し、受講者とのインタラクションを主体に進むプレゼンテーションに対応させたもの [52] などがある。これは、[41] のようなインタラクティブなフィードバック、[33] のようなスライド提示順の任意性（好きなスライドを好きな順番で表示）の機能をもちつつ、多くの教育場面で多用されている PowerPoint のプレゼンテーション資料を利用・流用することができる。

このような特徴をもつ教育向けプレゼンテーションソフトウェアが開発されているものの、これらはプログラミング言語教育用に特化したものではない。プログラミング言語教育用プレゼンテーションソフトウェアにおいては、手書き文字によるインタラクションよりも、サンプルコードを実行し、その結果を表示するインタラクションが重視される。プログラミング言語の動作はコンピュータ上によって行われるのであり、プログラムコードを人間が「代理的に」実行するよりも、コンピュータにより実行するほうが、自然である。

あるいは、Palette を用いることを考えた場合には、1 つのトピックに対して想定されるサンプルコードの実行結果を網羅的に用意しなければならないが、プログラミング言語の実行結果は、その想定される結果よりもはるかに多くの実行結果が起こりうるだろう（例えば、変数の値を変えた場合の実行結果など）。このことから、教師があらかじめ結果を想定したスライドを多数用意するよりも、サンプルコードの実行環境を用意し、その場でサンプルコードを編集でき、実行し、受講者に提示

するほうが合理的である。

現状では、それを行うために、プレゼンテーションソフトウェアと実行環境を別々に用意しているが、それらを統合的に利用することができるソフトウェアが必要とされる。

1.5.3　プログラミング言語教育とプレゼンテーションソフトウェア

本論文では、先述のように、プログラミング言語学習の初学者に対して、教師が一斉講義により講義を行う場面を想定する。

Buck・Stucki や Kölling・Rosenberg が主張するように、教師は初学者に対してサンプルコードを提示することは、コンピュータサイエンス教育の立場からも認められている [6][34]。インストラクショナルデザインの立場に立てば、新しい事項を提示したり、学習の指針を与える際にサンプルコードを提示するべきであるということについて、先に述べた。このような場合に、プレゼンテーションソフトウェアを用いると便利である。

このような授業形態を、より定式化することを考える。まずは、初学者に対するプログラミング言語教育について定式化する。初学者のプログラミング言語学習は、以下の各ステップをとると考えられる。

1. 知識の習得：基本的な文法や予約語などの、プログラミングを行う上で前提となる知識の習得。
2. コーディング：前項で得た知識を利用し、理解を確かめるために、実際に動作するプログラムコードを作成する。
3. 実行：前項で作成したプログラムコードを実行し、結果を確認する。
4. 検証：実行した結果が正しく動作していると考えられる場合には、次の項目に移る。そうでない場合には、なぜそのようになったのかを考え、その結果を基に再び 2 に戻る。

知識の習得は、9 教育事象における

- 学習者の注意を喚起する
- 学習者に目標を知らせる
- 前提条件を思い出させる
- 新しい事項を提示する

に当たる。学習者は、教師によって新規事項を提示され、知識として習得することになる。

コーディングは、

- 練習の機会をつくる

に当たる。前項で得た知識を確認するために、コーディングを行うという形で練習を行う。

実行・検証は、

- フィードバックを与える
- 学習の成果を評価する

に当たる。作成したコードを実行することで、学習者に対するフィードバックが与えられ、その成果が評価される。ただし、プログラムを実行しただけでは学習者（特に初学者）にとってフィードバックが十分でないことが多く、ここに教師の介在が必要である。

なお、9 教授事象の「保持と転移を高める」については、講義をはなれて、より発展的で大規模な応用演習問題を提示する方法によることを想定した。

ここに挙げたステップは、講義におけるプログラミング言語学習を単純化したものであり、初学者の理解が進むにつれてこれらのステップは必ずしも成り立たなくなるが、本稿では初学者向けのプログラミング学習を取り扱うので、これらのステップをもとに議論する。

これらのステップを、一斉講義において当てはめると、以下のような支援が必要である。

a 知識の習得に対し、適切な視聴覚教材・説明文を提示することによる支援。

b コーディングの実例（サンプルコード）を提示することによる支援。

c サンプルコードの実行結果を提示する支援。

d 実行した結果をどのように検証するべきかを口頭や視聴覚教材により伝える。場合によっては、受講者の成功例/失敗例を例示することによる検証の支援。

このようなステップごとの支援をまとめると、図 1.2 のようになる。

このような授業を支援するために、多くの場合にプログラミング開発・実行環境（Web 言語の場合、DreamWeaver や Aptana Studio など。以下、単に開発・実行環境と呼ぶ）やプレゼンテーションソフトウェアが利用される。例えば、プレゼンテーションソフトウェアは a や b に役立ち、実行環境は b、c、d に役立つと考えられる。すなわち、教師は、プレゼンテーションスライドで説明を行うが、サンプルコードの実行結果を表示する場合には画面を切り替えて実行環境を表示するといった授業手法が考えられる。

しかしながら、この手法には、先に述べたような「分裂注意効果」が発生する可能性があることが、問題点として考えられる。

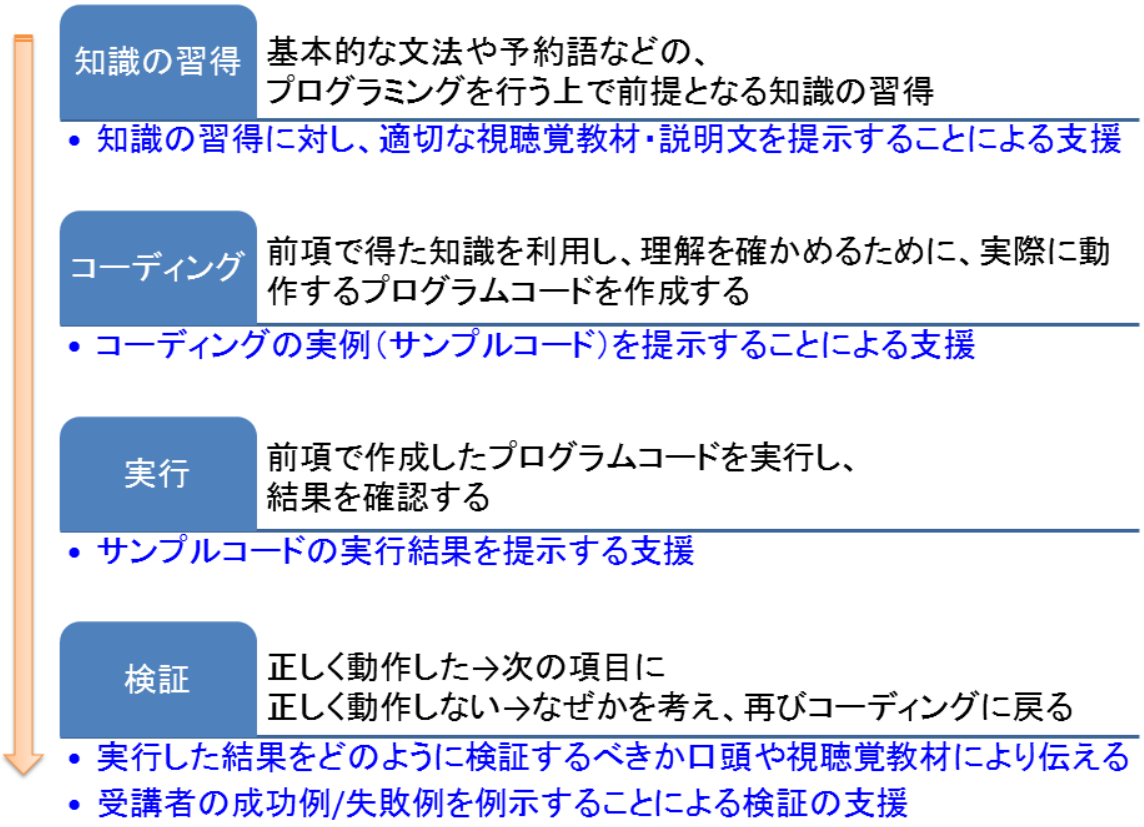

図 1.2　初学者のプログラミング学習のステップとそれに対する支援

1.6　プレゼンテーションの作成

1.6.1　既存のプレゼンテーション作成ソフトウェア

プレゼンテーションを行うためには、スライドを表示する機能が必要である一方で、スライドを作成・編集する機能が必要である。

PowerPoint や Google ドライブのプレゼンテーション、Prezi[26] には、スライドを作成・編集するための機能が用意されている。これらは、GUI（Graphical User Interface）による WYSIWYG（What You See Is What You Get）にスライドおよびプレゼンテーションの作成・編集を行うことができる。（詳細は、第 4 章で述べる。）

これに対して、テキストベースで編集を行うことを前提としたプレゼンテーションソフトウェアも存在する。例として、W3C HTML Slidy[8]、LaTeX Beamer[44] などが挙げられる。これらのソフトウェアを用いてプレゼンテーションを作成する場合には、HTML や LaTeX などのテキストベースで編集したファイルを、スライド形式での表示の確認をする必要がある。LaTeX Beamer ではプレゼンテーションファイルを論文形式などの別の表示形式のファイルを出力することができる [44] など、プレゼンテーションファイルを他メディアへ変換・転用する上で利便性が高いが、図形を多用する場合などに数値による修正が必要になるなどの欠点もある。

1.6.2　分裂注意効果をもたらさないためのプレゼンテーション作成

本論文では、プログラミング言語講義において、分裂注意効果をもたらさないために、次の 2 点が必要であると考えた。

- スライド内の表示要素を自由に配置できること
- スライド内に配置した要素が連携した表示を行うことができること

ここで、スライド内の表示要素とは、箇条書きなどの文章情報、サンプルコード、その実行結果、あるいは図表などのスライド内に表示されるものそれぞれを指す。

前者をみたすために、GUI による WYSIWYG なプレゼンテーション作成が行えることが必要と考えた。これは、テキストベースによる配置では、表示要素の微調整を視覚的に行うことができず、最適な表示を行うために多大な時間と労力を必要とするからである。言い変えるならば、スライド内に要素を自由に配置できることにより、教師が分裂注意効果の影響を軽減したスライドを作成できることにつながると考えた。

後者をみたすため、プログラミング言語講義においては、サンプルコードとその実行結果が対応していること、サンプルコードに修正を加えた場合、その修正結果が実行結果に反映されることが必要であると考えた。これらの対応がない場合、時間的あるいは空間的な分裂注意効果が発生するものと考えられるからである。

本論文で開発した CodEx システムは、W3C HTML Slidy をベースとしている。後述する第 2・3 章で述べる「CodEx」では、HTML ベースの編集が必要であるが、

第4・5章で述べる「CodEx GUI Editor」の開発により、GUIによるWYSIWYGなプレゼンテーション作成を行うことを可能とした。

1.7　研究の目的

本論文では、講義形式によるプログラミング教育をより効率的に行うためのプレゼンテーションシステム「CodEx」を提案し、その有効性を実証することを目的とした。

プレゼンテーションシステムは、「スライドを表示し、プレゼンテーションするための機能」と、プレゼンテーションソフトウェアを用いて講義を行う教師の準備を支援するため、「スライド・プレゼンテーションを作成・編集するための機能」が必要である。本論文では、前者の実装としてCodEx、後者の実装としてCodEx GUI Editorを開発した。前者の開発については第2章で、後者の開発については第4章で述べる。

また、それぞれの有効性を検証するための評価実験を行った。この評価実験については順に第3章、第5章で述べる。

インストラクショナルデザイン・Worked Exampleの原則と、本研究で開発されるシステムの要求仕様を表した図を図1.3に示す。次章以降では各原則の項目に対して要求仕様を定義し、システムの開発を行う（第2・4章）。さらに、その要求仕様を満たしているかを、ユーザビリティ評価により実証する（第3・5章）。

最後に、模擬授業を用いた実験により、CodExによる学習効果について検討する（第6章）。

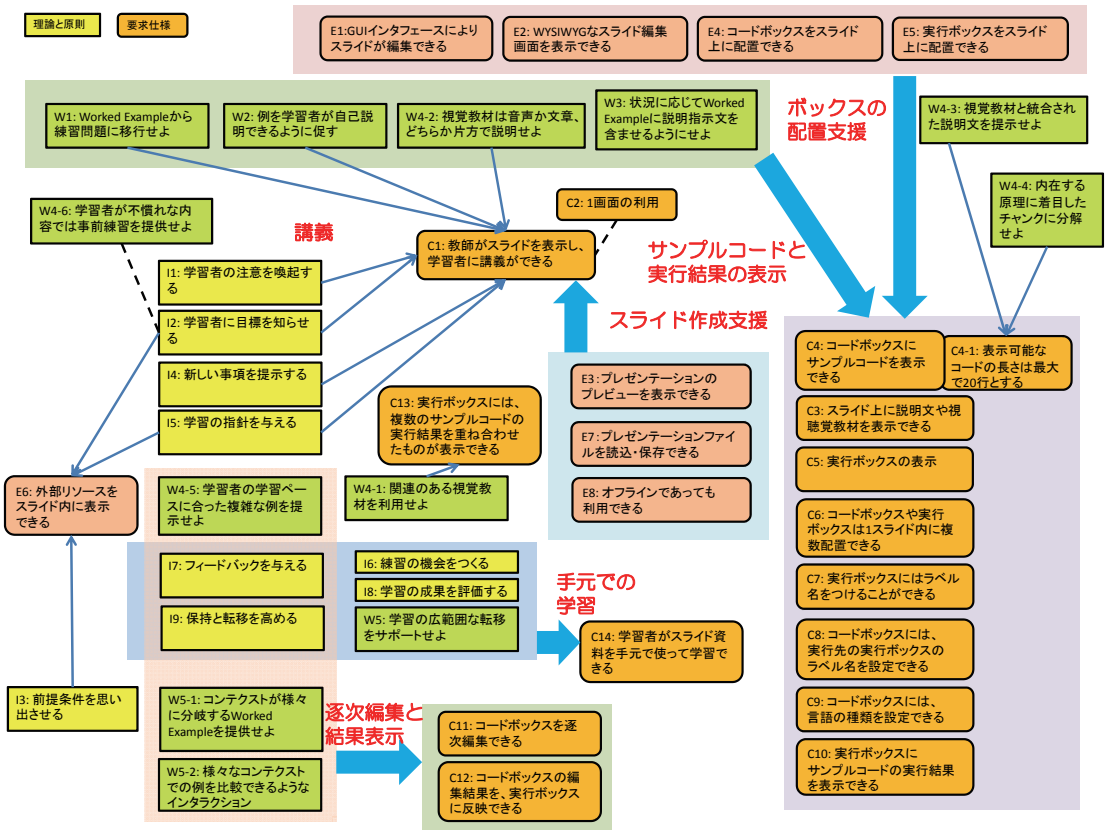

図 1.3　インストラクショナルデザイン・Worked Example の原則とシステムの要求仕様

1.8　用語の定義

本節では、本論文で用いた用語の定義について示す。

プレゼンテーションソフトウェア：プレゼンテーションファイルを作成、表示、操作するソフトウェア。プレゼンテーション作成ソフトウェア、プレゼンテーション表示ソフトウェア、プレゼンテーション操作ソフトウェアなどの場合はそれぞれ、作成、表示、操作のみの機能を持っているソフトウェアを指す。

スライド：講義時にスクリーンに表示する資料一枚一枚のことを指す。

プレゼンテーション（名詞）：講義時に表示する資料全体を指す。

プレゼンテーションする（動詞）：資料を用いて講義を行う、あるいはスライドを表示しながら説明すること。

Web プログラミング言語：本稿では、HTML･CSS（Cascading Style Sheet）･JavaScript について取り扱うため、これらの言語のことを指す。また、これらのコードを記述することを「Web プログラミング」と呼ぶことにする。

なお、Web プログラミングを行う際には、他にサーバサイドで動作させる言語や、Flash などのプラグインを必要とする言語もあるが、前者は初学者にとってサーバ構築などの負担が大きいことや、後者は HTML5 での置き換えが進展していることなどを考慮し、本研究では除外した。

講義：教師が学生の前に立ち、説明・指示・演示等の話を行うことにより学習を進める教育形態を指す。

授業：教育機関において、特定の部屋に集合し、一斉に学習すること。上記の講義に加え、プログラミング言語教育においては演習なども含まれる。

以下は、本論文で開発したソフトウェア群の名称である。

CodEx：プログラミング言語教育用プレゼンテーションを表示するソフトウェア。実体としては、HTML ファイルと付属ファイルからなり、該当する HTML ファイルをブラウザで開くことで、プレゼンテーションを行うことができる。CodEx 単体では、プレゼンテーションの編集機能は備えていない。

CodEx GUI Editor (Editor)：CodEx によるプレゼンテーションを作成･編集するためのソフトウェア。実体としては HTML ファイルであり、該当する HTML ファイルをブラウザで開くことで、CodEx プレゼンテーションファイルを作成･編集することができる。本文中では、単に Editor と呼んでいる箇所がある。

CodEx システム：上記の CodEx と CodEx GUI Editor を含め、このように呼ぶこととする。

1.9　論文の構成

本論文の構成について述べる。

本論文では、目的を達成するために、システム開発とその機能評価、さらにシステムにより実現される授業の評価を行った。具体的には、「スライド表示部」と「スライド編集機能」に大別される機能の用件定義・実装とその評価を行った。それぞれ、文献 [22] と [58] に対応する。本論文では、前者を第 3・4 章に、後者を第 5・6 章にて詳細に述べている。さらに、第 6 章で CodEx を用いた講義の評価を行った。そのほか、各章の内容をまとめると以下のようになる。

- 第 1 章

  本章では、本研究の序論として、本研究の研究背景について述べた。プログラミング言語教育の支援や、インストラクショナルデザイン、マルチメディアラーニング理論について述べ、本論文の位置付けを明確にした。その上で、本研究の目的を述べた。

- 第 2 章

  第 2 章では、Web プログラミング言語教育のためのプレゼンテーションソフトウェア「CodEx」の開発について述べる。

- 第 3 章

  第 3 章では、CodEx プレゼンテーションソフトウェアを講義に用いることを想定し、評価した実験について述べる。評価実験としては、「教師によるユーザビリティ評価」「CodEx を用いた模擬授業による学生の評価」の 2 つについて述べている。

- 第 4 章

  第 4 章では、「CodEx GUI Editor」の開発について述べる。

- 第 5 章

  第 5 章では、CodEx GUI Editor により、教員がスライドを容易に作成可能であるかどうかを確認するために行ったユーザビリティ評価について述べる。

- 第 6 章

  第 6 章では、CodEx システムにより実現される授業の 1 例についての、模擬授業による評価実験について述べる。

- 第 7 章

  第 7 章では、本論文の内容をまとめ、結論を述べる。

  これらの構成を図 1.4 に示す。

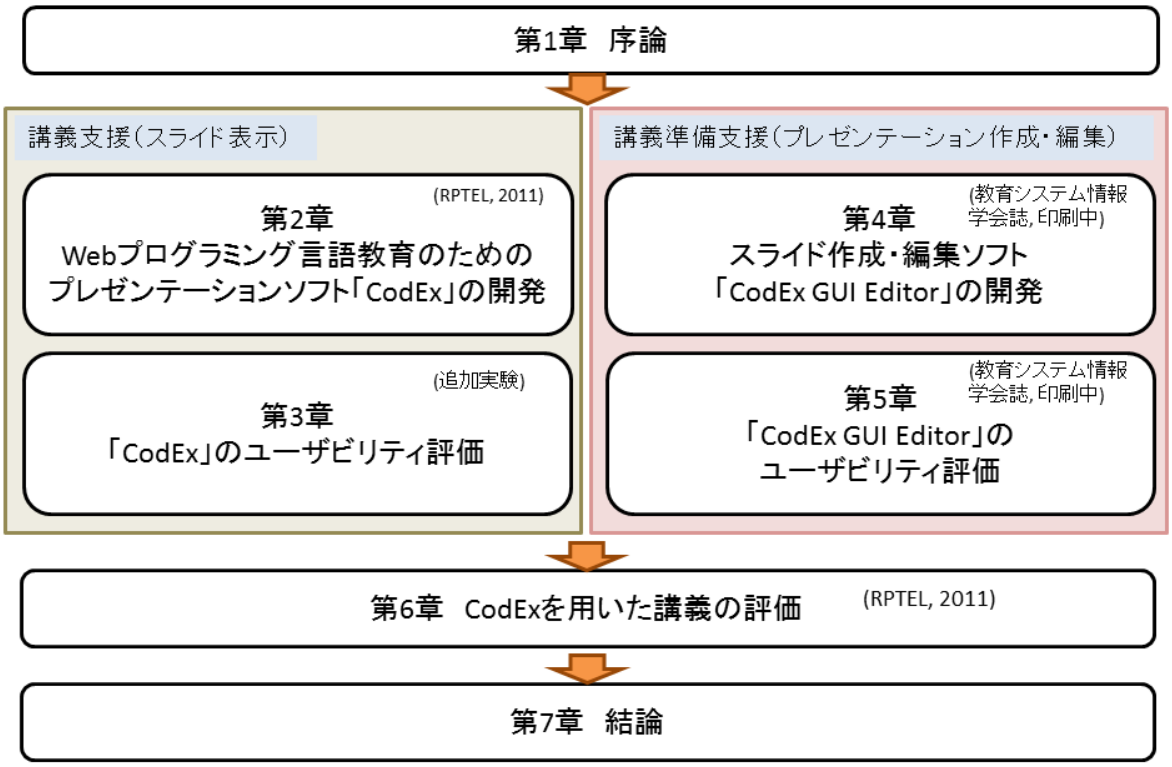
第1章　序論
講義支援（スライド表示）
(RPTEL, 2011)
第2章
Webプログラミング言語教育のための
プレゼンテーションソフト「CodEx」の開発
(追加実験)
第3章
「CodEx」のユーザビリティ評価
講義準備支援（プレゼンテーション作成・編集）
(教育システム情報
学会誌, 印刷中)
第4章
スライド作成・編集ソフト
「CodEx GUI Editor」の開発
(教育システム情報
学会誌, 印刷中)
第5章
「CodEx GUI Editor」の
ユーザビリティ評価
第6章　CodExを用いた講義の評価
(RPTEL, 2011)
第7章　結論

図 1.4　本論文の構成

1.10　本章のまとめ

本章では、本研究の序論として、本研究の研究背景について述べた。プログラミング言語教育の支援や、インストラクショナルデザイン、マルチメディアラーニング理論について述べ、本論文の位置付けを明確にした。その上で、本研究の目的を述べた。

第2章

Webプログラミング言語教育のためのプレゼンテーションソフトウェア「CodEx」の開発

本章では、Web プログラミング言語教育のためのプレゼンテーションソフトウェア「CodEx」の開発について述べる。

2.1　CodEx の概要

本ツールは、Web プログラミング言語の実行環境（サンプルコードの編集および実行結果の表示）機能を備えた Web ブラウザベースのプレゼンテーションツールである。本ツールを用いた場合の画面構成を図 2.1 に示す。図 2.1 に示すように、CodEx のスライドの特徴として、「コードボックス」と「実行ボックス」を表示できることが挙げられる。

2.2　HTML プレゼンテーションの利点と欠点

CodEx の実装には、HTML ファイルを用いたプレゼンテーション（以下、HTML プレゼンテーションと呼ぶこととする）である W3C HTML Slidy[8] の機能を拡張している。そのため、CodEx は HTML プレゼンテーションの特徴をもつ。

以下、HTML プレゼンテーションの利点と欠点について述べる。

2.2.1　HTML プレゼンテーションの利点

HTML プレゼンテーションは、一般に、以下の利点をもつ。

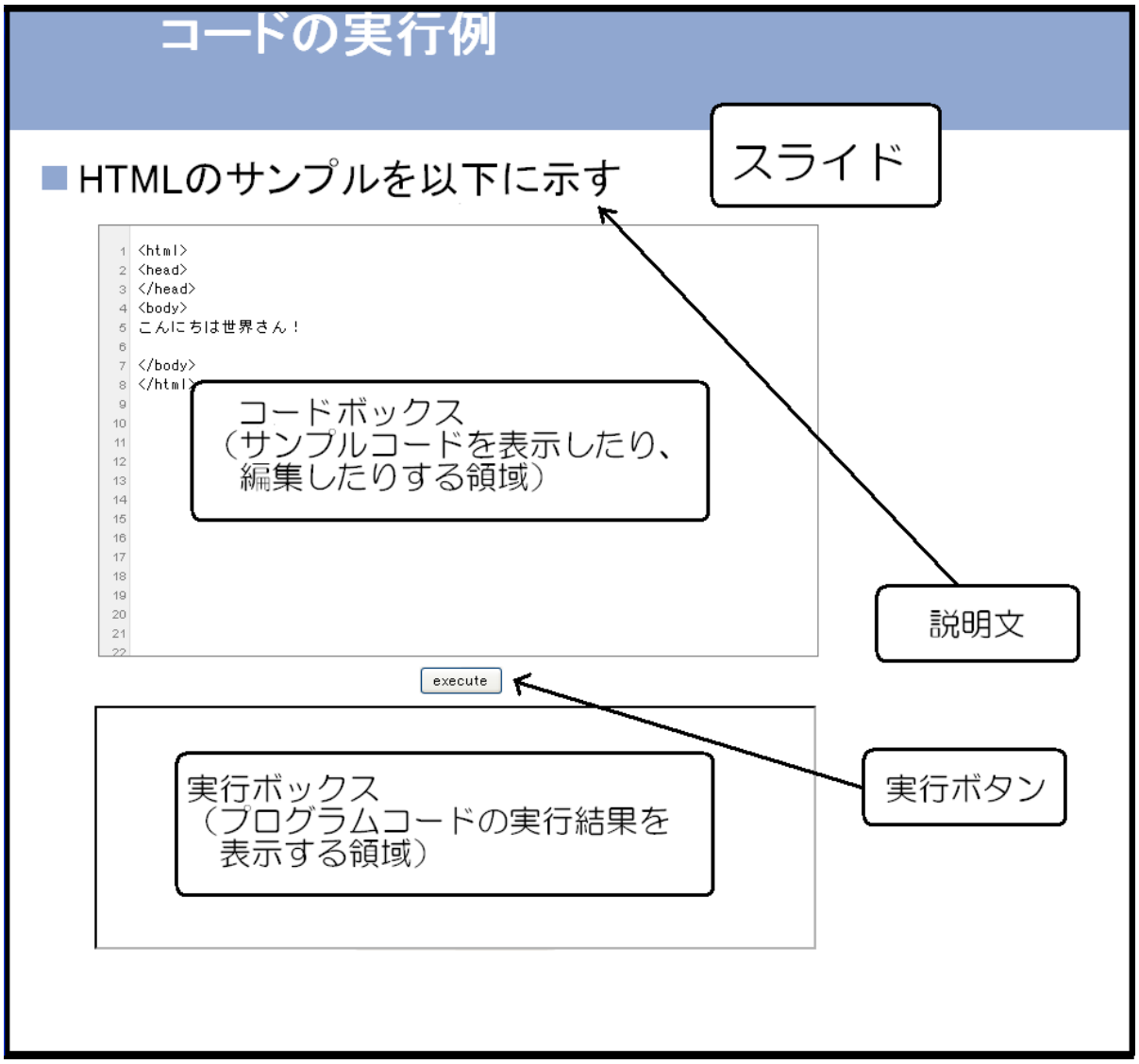

図 2.1　CodEx のスライド例

デバイスの汎用性

HTML を用いたプレゼンテーションファイルは、インターネット・ブラウザを用いて見ることができる。PC だけでなく、スマートフォンやタブレット端末が普及しつつある現在においては、どのデバイスでも見ることができる点は大きな長所と考えられる。

HTML5/CSS3 による最新技術の利用

HTML5/CSS3 の技術は、日々新しい機能が取り入れられ、進歩している。例えば、Kaminishi らは、HTML5 で取り入れられた WebRTC の技術を用いて Web カメラをスライド上で表示するなどの機能をもったプレゼンテーションツールを開発している [24]。

論理性の保持

適切に記述された HTML ドキュメントは、文書構造や表示される要素の役割などの論理性をもっている。これを利用することで、例えば必要な要素を拡大表示するなどの処理を行うことができる [23]。

Web 上のコンテンツの利用

適切な著作権管理を前提とすれば、Web 上のリファレンス等をスライド内に引用表示したり、YouTube などの動画コンテンツを提示したりすることができる。

2.2.2 HTML プレゼンテーションの欠点

スライド作成時のレイアウト設定

HTML ドキュメントは、一般に、文書の内容と論理構造を記述するための HTML、ドキュメントの見た目を設定するための CSS、ドキュメントにおける動作を記述するための JavaScript などからなる。Slidy などの HTML プレゼンテーションツールについても同様である。このうち、各スライド内の要素のレイアウトを決定するのは CSS である。通常、CSS の設定はテキストベースで行うが、HTML の要素の大きさや配置など、とくに数値を設定する項目についてテキストベースで行うことは困難を伴う。以下、そのような例として、スライド内の任意の場所に画像などを配置する場合について述べる。

一般の HTML において、HTML に記述された要素は、デフォルトで他の要素に対して相対的な位置に配置される（CSS における position: static または position: relative）。一方で、スライドを作成する際には、画像などの要素を各スライド内での絶対的な位置に配置できると便利である。このような配置を行うためには、CSS

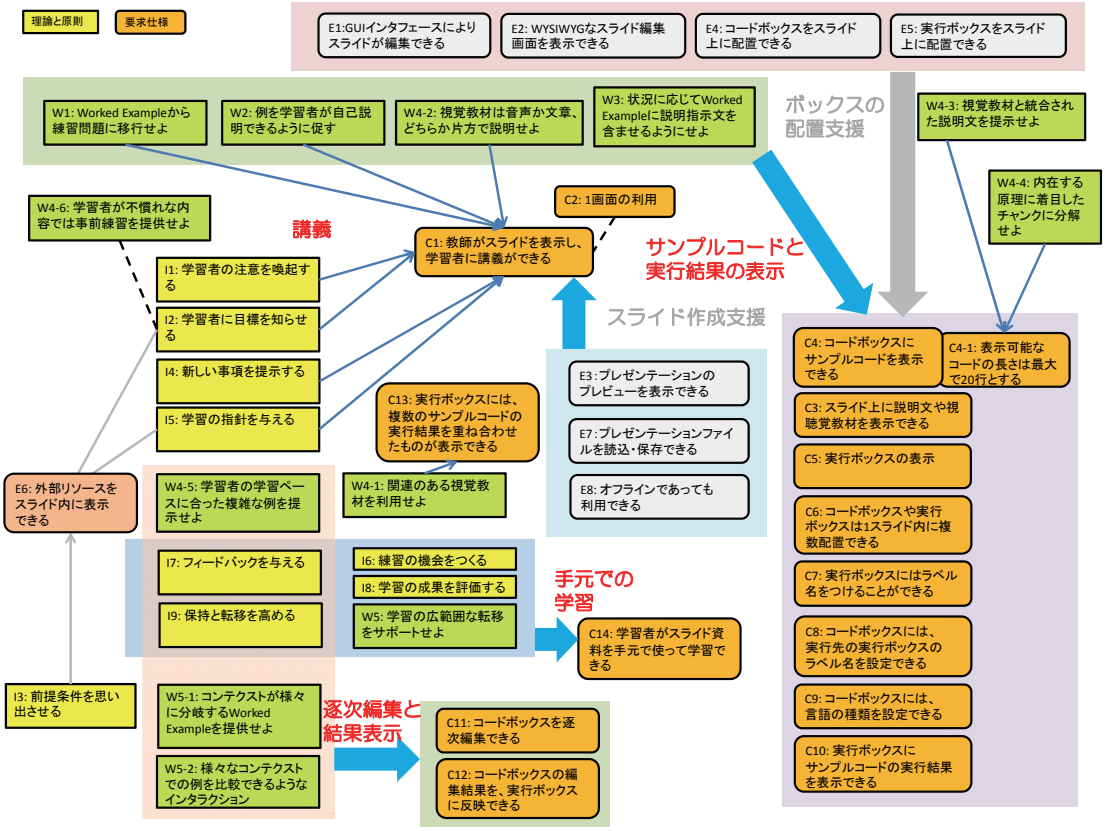

図 2.2　CodEx の要求仕様の位置づけ

にて position: absolute と指定すればよいが、配置する場所を座標の数値で設定する必要がある。この座標の位置をテキストベースで行うのは難しい。なぜならば、数値による位置指定では、配置を直観的に確認できないからである。そのため、このような配置の決定を容易にするためには、WYSIWYG なインタフェースで行うことができるのであれば有用であると考えられる。この欠点の解決策については、第 4 章で述べる。

2.3　CodEx の要求仕様

開発の背景や前述の HTML プレゼンテーションの利点と欠点を踏まえて、CodEx の要求仕様を挙げる。要求仕様とインストラクショナルデザイン・Worked Example の原則との対応・位置づけを、図 2.2 に示す。図の着色された部分が、CodEx の要求仕様である。図の灰色部分は、第 4 章・第 5 章で述べる範囲である。

以下、要求仕様の各項目について説明する。

2.3.1 C1: 教師がスライドを表示し、学習者に講義ができる

プレゼンテーションソフトウェアの前提として、スライド資料が表示できることが挙げられる。プレゼンテーションは、順序を持った複数のスライドの集合からなり、その中の 1 つのスライドがスクリーンに表示される。ユーザは、スライドをボタン操作により切り替えられる。ボタンを押すと、ひとつ後あるいはひとつ前のスライドが表示される。

教師は、これを利用して、講義を行う。

この仕様により、「I1: 学習者の注意を喚起する」「I2: 学習者に目標を知らせる」「I4: 新しい事項を提示する」「I5: 学習の指針を与える」「W2: 例を学習者が自己説明できるように促す」「W4-2: 視覚教材は音声か文章、どちらか片方で説明せよ」が達成されることが予想される。

2.3.2 C2: 1 画面の利用

本研究では、スクリーン 1 面が用意された講義環境を想定する。プログラミング言語教育における先行研究には複数のスクリーンを用いたものもあるが [46][42]、設備的な問題から複数スクリーンを用いずに講義環境を構築することを前提とする。

2.3.3 C3: スライド上に説明文や視聴覚教材を表示できる

プレゼンテーションソフトウェアを用いて講義を行う場合、一般に箇条書きなどの説明文や画像などの図表が用いられる。これらが表示できるのはもちろんのこと、スライド上に、動画や既存の Web ページといった視聴覚教材を表示することができるとよい。たとえば、Web リファレンスサイトを提示することで学習内容の見通しをよくしたり、授業のイントロダクションとして動画を提示したりすることが考えられる。

2.3.4 C4: コードボックスにサンプルコードを表示できる

スライド上に、「コードボックス」と呼ぶサンプルコードを表示できる領域を表示することができる。このコードボックス上には、HTML/ CSS/ JavaScript のコードを表示することができる。コードはシンタックスハイライトにより色づけされて表示される。

コードの行数が長い場合には、ボックスの大きさや、表示されているフォントの大きさは変化せず、コードが部分的に表示される。表示される部分は、スクロールバーを利用して変更することができる。

C4-1: 表示可能なコードの長さは最大で 20 行とする

本研究では初学者向けの講義を想定していることから、長い行数のコードは表示しないことを想定している。具体的には 20 行を目安とした。この行数は、「W4-4: 内在する原理に着目したチャンクに分解せよ」「W4-6: 学習者が不慣れな内容では事前練習を提供するか、学習者が慣れた内容を利用せよ」を満たすために、必要十分なものと判断し、設定した。なお、この行数の議論については第 3 章で述べる。

2.3.5 C5: 実行ボックスの表示

コードボックス上に表示したコードを実行した結果を表示するための領域「実行ボックス」を表示することができる。実行ボックスには、初期状態で何も表示されていない。

2.3.6 C6: コードボックスや実行ボックスは 1 スライド内に複数配置できる

コードボックスや実行ボックスは 1 スライド内に複数配置できる。

コードボックスを複数配置できることで、複数のコードを重ね合わせて実行した結果を示すことができる。とくに、Web プログラミング言語の場合には、HTML/CSS/JavaScript といった複数種類の言語を組み合わせて最終的な表示結果を得る必要があるため、コードボックスを複数配置できることが重要である。

実行ボックスを複数配置できることで、異なるサンプルコードの実行結果の比較を行うことができる。

2.3.7 C7: 実行ボックスにはラベル名をつけることができる

前項で述べたように、コードボックスと実行ボックスは、ひとつのスライドの中に複数配置される。これらを対応づける手段が必要である。

CodEx では、プレゼンテーションファイルの HTML ソース上で、実行ボックスに HTML の id 属性を指定する。さらに、コードボックス側で、サンプルコードの結果を表示させたい実行ボックスを指定する。

そのため、実行ボックスには id によるラベル名をつけることができる必要がある。

2.3.8 C8: コードボックスには、実行先の実行ボックスのラベル名を設定できる

前項で説明したように、コードボックスと実行ボックスの対応付けを行う必要がある。

コードボックス側で、サンプルコードをの結果を表示させた実行ボックスを指定することができる必要がある。

2.3.9 C9: コードボックスには、言語の種類を設定できる

コードボックスには、HTML/ CSS/ JavaScript といった複数種類の言語で書かれたサンプルコードを表示することができる。
これらを区別することができなければならない。

2.3.10 C10: 実行ボックスにサンプルコードの実行結果を表示できる

実行ボックスには、コードボックス上に表示したサンプルコードを実行した結果を表示することができる。
実行ボックスがスライド内に表示できることにより、サンプルコードの実行結果を同一のスライド内で確認することができる。
実際に表示させるには、スライド上に表示されたボタンをクリックすることで表示できるようにする。
これら、C3〜C10 の仕様により、「W1: Worked Example から練習問題に移行せよ」「W2: 例を学習者が自己説明できるように促す」「W3: 状況に応じて Worked Example に説明指示文を含ませるようにせよ」「W4-1: 関連のある視覚教材を利用せよ」「W4-5: 学習者の学習ペースに合った複雑な例を提示せよ」「W4-6: 学習者が不慣れな内容では事前練習を提供するか、学習者が慣れた内容を利用せよ」が達成されることが予想される。
なお、実行を行うときに、サンプルコードに誤りがありエラーが発生する場合には、エラーがあることを表示する。

2.3.11 C11: コードボックスを逐次編集できる

コードボックスの内容は、スライドを表示した状態であっても PC のキーボードを用いて編集することができる。
この機能により、教師が説明を行っている最中であっても、サンプルコードを変化させて表示することができる。「コードボックスの編集結果を、実行ボックスに反映」とあわせて、教師が受講者の反応を受けたインタラクティブな授業を行うことができる。

2.3.12 C12: コードボックスの編集結果を、実行ボックスに反映できる

上述のように、コードボックスは逐次編集できるが、編集した結果を実行ボックスに反映することができる。反映する際には、スライド上（コードボックスの下）に配置されたボタンをクリックする。

C11、C12 の仕様により、「I7: フィードバックを与える」「W4-5: 学習者の学習ペースに合った複雑な例を提示せよ」「W5-1: コンテクストが様々に分岐する Worked Example を提供せよ」「W5-2: 学習者がアクティブに様々なコンテクストでの例を比較できるようなインタラクションを提供せよ」が達成されることが予想される。さらに、学習の転移が進むことから、「I9: 保持と転移を高める」を満たすことにもつながる。

2.3.13 C13: 実行ボックスには、複数のサンプルコードの実行結果を重ね合わせたものが表示できる

前述のように、Web プログラミング言語の場合には、HTML/ CSS/ JavaScript といった複数種類の言語を組み合わせて最終的な表示結果を得る必要がある。そのため、複数のコードボックスの実行結果を重ね合わせた結果を表示できることが必要である。

関連のある情報を複数用いて学習することから、「W4-1: 関連のある視覚教材を利用せよ」ことに役立つと考えられる。

2.3.14 C14: 学習者がスライド資料を手元で使って学習できる

スライド資料を学生が手元の PC で見ることができる。さらに、学生が手元の PC でサンプルコードを実行し、実行結果を確かめることができる。加えて、学生がサンプルコードを編集し、編集後の結果がどのようになるかも確かめることができる。

この仕様により、「I6: 練習の機会をつくる」「I7: フィードバックを与える」「I8: 学習の成果を評価する」「I9: 保持と転移を高める」ほか、受講者に、実際に様々な例を経験させることで「W5: 学習の広範囲な転移をサポートせよ」を行う上で重要である。

2.4 CodEx の構成要素

以下、CodEx を構成する要素について説明する。

2.4.1 コードボックス

「コードボックス」は、サンプルコードを表示したり、編集したりする領域である。コードボックスを用いることで、受講者に対してサンプルコードを提示することができる。本ツールでは、サンプルコードはシンタックスハイライト（コードへの色づけ）を行った状態で表示することができる。また、サンプルコードは授業中に編集することもできる。これにより、スライドの初期状態に全ての事項を記述せず、説明を進めながらサンプルコードを完成させていくといった利用法もできる。また、受講者からの質問に対し柔軟に対応することも可能である。

2.4.2 実行ボックス

実行ボックスは、コードボックスに入力されているコードを実行した結果を表示する領域である。「コードボックス」に記述されたサンプルコードがどのような結果をもたらすのかを、同じスライド内に用意した「実行ボックス」に表示することができる。また、サンプルコードは、他のアプリケーションを利用することなく、スライドショーを行いながらであっても編集することができる。さらに、編集したコードの実行結果をワンクリックで実行結果に反映させることができる。

2.5 CodEx の機能

本節では、ツール上で Web プログラミング言語を実行可能にするために用意された機能を説明する。

2.5.1 スライド内に表示される要素など

コードボックス

スライド内に編集可能なテキストエリアである「コードボックス」を表示することができる。それぞれのコードボックスに対して、実行したい言語を指定することができる。（現在、HTML/ CSS/ JavaScript に対応している。サーバを別途用意すれば、PHP にも対応可能である。）コードボックス内に表示されるコードは、スライドショーを開始した時点では、あらかじめ HTML ファイルに記述されたコードが表示される。その一方で、スライドショーを行いながら、表示されているコードに変更を加え、編集することもできる。

実行ボックス

実行ボックスは、コードを実行した結果を表示する部分である。実行ボックスは iframe により構成されている。言いかえるならば、ボックス内に表示されているの

は、実行結果を表示するために用意された実行結果表示用の HTML である。実行ボックスを示す要素には、id 属性が指定されている必要がある。これは、実行ボックスを一意に区別するためにつけるものである。コードボックス側では、ボタンを実行した際に呼び出される関数の引数として、表示先となる実行ボックスの id を指定する。これは、コードボックスに書かれたコードの実行結果をどこで表示するかを指定するものである。このようにして、ユーザは実行ボックスとコードボックスを関係づける。

実行コードは、実行結果表示用 HTML 上で実行・表示される。コードを実行するためのライブラリとして、実行結果表示用 HTML は exectext.js という JavaScript ファイルを読み込むようにしておかなければならない。このファイル内には、DOM（Document Object Model）操作などにより各言語を実行するための関数などが記述されている。

実行結果表示用 HTML は教師があらかじめ用意しておく。iframe には外部ドメインの Web ページを設定することもできるが、実行結果表示用 HTML としては利用できない。セキュリティの関係上、スライドファイルと同一のディレクトリに置かれた HTML ファイルのみで利用できることに注意しなければならない。通常の利用では、CodEx ライブラリ内に組み込まれた、実行ボックスに表示させるための HTML ファイルを表示することを想定している。この HTML ファイルを読み込んだ場合の実行ボックスの挙動としては、以下のようになる。

1. コードの実行前には、背景色がついた状態。これにより、コードが実行されていないことを示す。

2. コードを実行後には、一時的に背景色を消す（通常のブラウザの挙動では、デフォルトの背景色である白色が表示される）。その後、コードの実行結果を表示する。

コード実行ライブラリ

コードボックス内に記述された実行コードの内容は、実行ボックス内に表示されている実行結果表示用 HTML に送られる。実行結果表示用 HTML には、前述のように exectext.js が読み込まれている。送られてくるコードが HTML/CSS/JavaScript それぞれの場合に応じた関数が呼び出され、結果が表示（実行）される。

実行ボタン

コードボックスの下には、コードを実行するための実行ボタン（execute）が用意されている。実行ボタンを押すと、指定されたラベルをもつ実行ボックスにてそのコードボックス内のコードが実行される。コード実行機能により、スライド内でサンプルコードを編集でき、かつ実行可能なプレゼンテーションツールを実現した。

2.5.2 各種コードの実行

以下で、各言語を指定した場合の動作モードについて説明する。なお、本稿では HTML を表示させることや CSS を適用することについても「実行」という用語で統一する。

HTML モード

src 属性に html を指定した場合、実行コードを HTML として表示させた結果を実行ボックス内で見ることができる。すでに実行ボックス内に HTML が表示されている場合は、その表示内容を消去して新たな結果が表示される。（追記されるわけではない。）実行結果表示用 HTML には、exectext.js が読み込まれている。実行結果表示用 HTML は、コードボックスからコードを受け取り、exectext.js 内の関数を利用して結果を表示する。

また、実行コードとしては、HTML 全体を記述する場合（head タグや body タグを含む記述）、部分的に記述した場合（これらを含まない場合）の双方に対応している。実行結果表示用 HTML には、exectext.js が読み込まれている。実行結果表示用 HTML は、コードボックスからコードを受け取り、exectext.js 内の関数を利用して結果を表示する。渡されたコードがどちらの形式であるかは exectext.js 内の関数によって自動的に判別される。サンプルコード内に HTML 全体を記述することなく、演示するのに必要な部分のみを記述することができるように考慮している。

HTML コードを実行する際には、まず実行コードの内容について解析する。実行コードが head タグや body タグを含む場合・含まない場合に分けて処理される。

まず、実行コードが body タグを含まない形式の場合について説明する。実行ボックス内に表示された HTML の DOM 操作を行い、body タグの innerHTML プロパティの値を実行コードで置き換える。すなわち、もともと実行ボックス内に表示されていた HTML の head タグや body タグによって、ページを構成するのに必要なタグが補完される。body タグを含んだ実行コードの場合は、body タグ内に記述された部分のみを取り出す。その後、body タグを含まない場合と同様の処理を行う。この様子を図 2.3 に示す。head タグの内容については、必要な情報を取り出し、ページに反映させている。実行コード内に title タグがある場合には、DOM 操作により実行結果表示用 HTML の既存 title タグの内容を置き換える。また、実行コード内に style タグがある場合には、その記述内容を下に示す CSS の実行関数に渡す。

CSS モード

実行ボックス内に表示されている HTML に対して、CSS を適用する。execute ボタンを押すと、実行ボックス内の表示に変化を与えることができる。この場合も、既存の CSS 情報は新たに実行される CSS の記述による表示に置き換わる。（HTML モード同様、こちらも追記ではない。）実行ボックス内に表示する実行結果表示用 HTML（CSS を適用する HTML）は、ユーザがあらかじめ用意した HTML ファイ

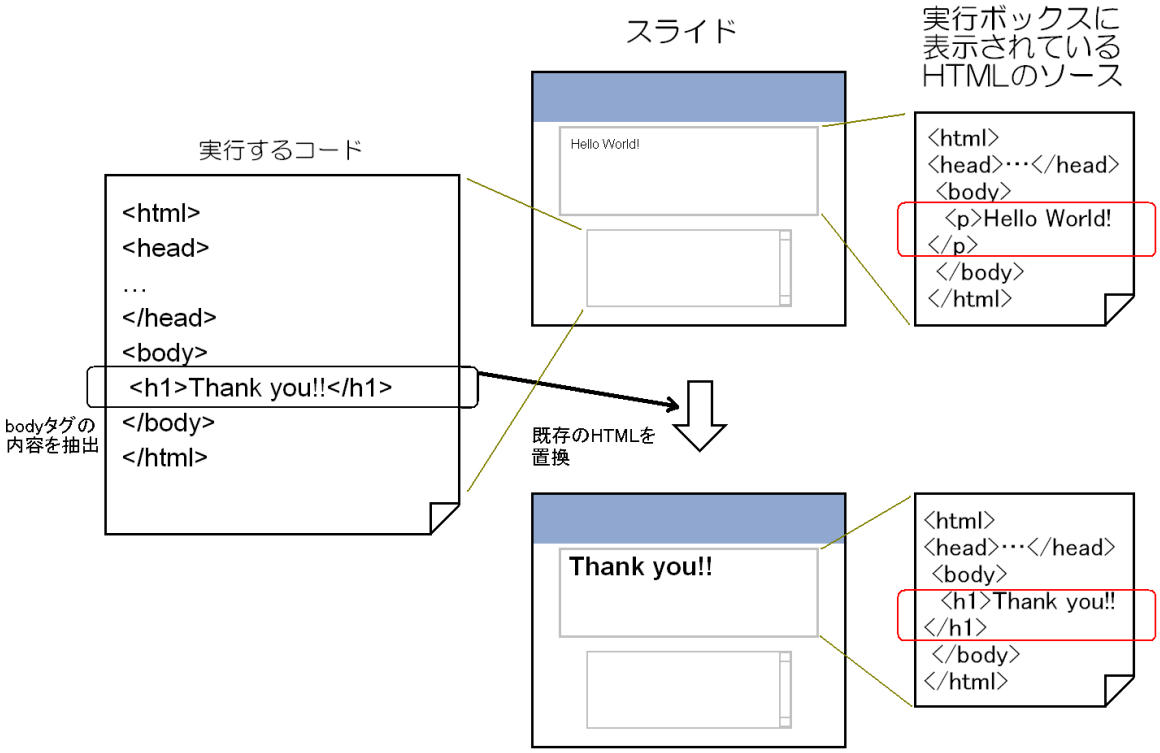

図 2.3　body タグを含む実行コードを実行ボックスに表示

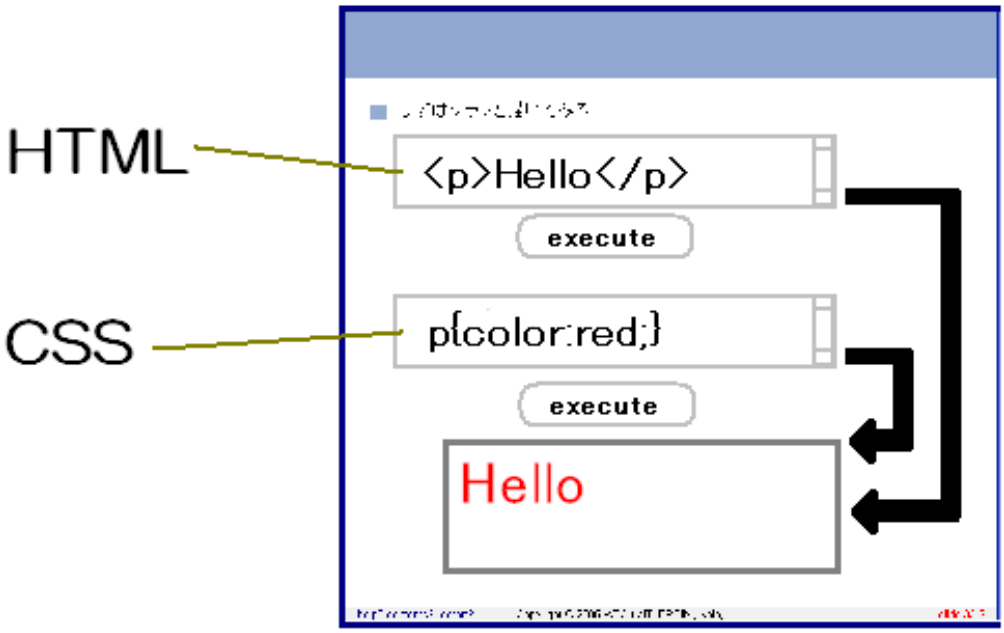

図 2.4　1 つの実行ボックス内で HTML と CSS を実行する例（別のコードボックスに書かれたコードをひとつの実行ボックスに表示することができる）

ル、又は HTML モードにてサンプルコードを実行することで表示させた HTML の両方に対応している。後者の場合は、同一のスライド内に HTML 用のコードボックスと CSS 用のコードボックスをそれぞれ用意した上で、同じ実行ボックスを実行先として指定する必要がある。1 つの実行ボックスが HTML と CSS の 2 つのコードボックスから指定されている例を図 2.4 に示す。

なお、CSS を HTML 内に埋め込んで利用する場合には、CSS モードではなく、HTML モードのコードボックスに記述する。excetext.js 内の関数によって行われる処理としては、実行結果表示用 HTML の既存 CSS を、実行コードで置き換える。exectext.js 内の関数は、DOM 操作により実行結果表示用 HTML の既存 style タグや link タグを削除し、実行コードの内容をもとに新たに定義した style タグを加える。

JavaScript モード

実行ボックス上で JavaScript コードを実行することができる。HTML や CSS のサンプルコードと組み合わせることも可能であるが、その場合は、同一のスライド内にそれぞれ独立したテキストエリアを用意する必要がある。図 2.5 に、JavaScript モードで実行した例を示す。図 2.5 では alert を表示しているが、これはスライドの HTML によって表示されたものではなく、実行ボックス内の実行結果表示用 HTML によって表示させられたものである。JavaScript モードの場合コードボックスに記述された実行コードを実行ボックス内の HTML に渡す。その上で、exectext.js 内の関数により、実行結果表示用の HTML 上に実行結果が表示される。具体的には、JavaScript の eval 関数を、受け取った実行コードの内容を引数として実行ボックス内の HTML にて実行している。

2.5.3 誤りを含むコードの実行

サンプルコード実行時のデバッグ支援のひとつとして「構文エラー表示機能」がある。この機能では、サンプルコードの結果を表示しようとした際に構文エラーがあると、それを指摘・表示することができる。

HTML の構文エラーに対する支援

HTML の構文エラー、たとえばタグの閉じ忘れ等を指摘し、表示する。例として、タグの閉じ忘れがある HTML コードを実行した場合のエラーの表示例を図 2.6 に示す。

HTML の文法チェックは、XHTML の文法に準拠する。これは、教育的な観点から厳密な文法チェックが必要であると考えられることと、実装上 HTML5 準拠によるチェックが難しいことが理由である。

JavaScript の構文エラーに対する支援

コード実行時において、発生したエラーの種類やコード中の間違いの位置を表示する。例として、シンタックスエラーを含む JavaScript コードを実行した場合のエラーの表示例を図 2.7 に示す。

CSS の構文エラーに対する支援

現時点では、CSS のエラーを取得することができない。解決策として、外部のプラグイン・アドオン等を用いるか、独自に CSS パーサを開発する方法が考えられる。

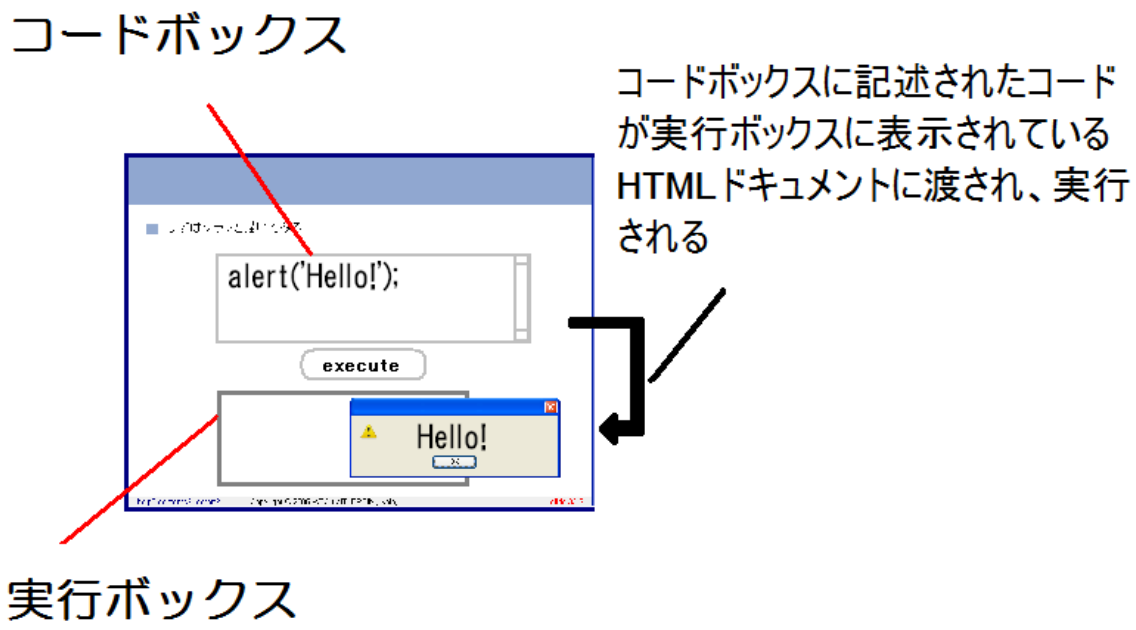

図 2.5　JavaScript の実行

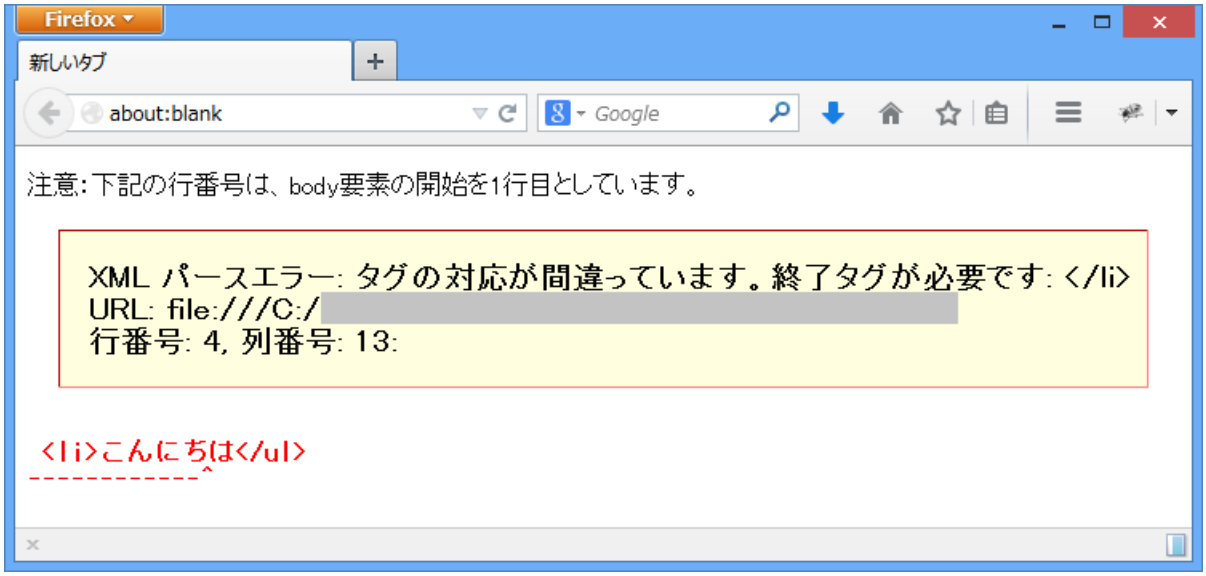

図 2.6　HTML コードでのエラー表示例（省略部にはファイルのパスが入る）

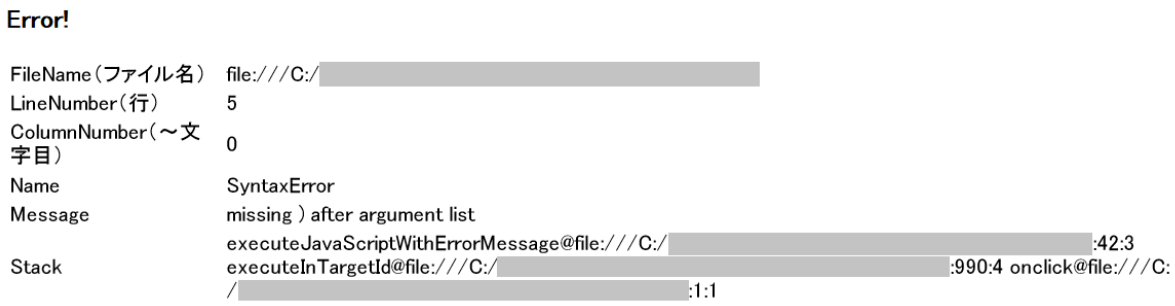

図 2.7　JavaScript コードでのエラー表示例（省略部にはファイルのパスが入る）

2.6　設計の方針

本ソフトウェアは、Web プログラミングに対する専門的知識を持った教師が利用することを想定しているので、機能制限については原則として行わないことを前提としている。例として、本ソフトウェアにおいては JavaScript でいわゆる無限ループの状態を作り出すことも可能である。本ソフトウェアは、教師が生徒に対してあえてその危険性を教える必要があることを考え、そのような制限を行っていない。また、HTML サンプルコードの実行は、既存の表示内容に付け加える形で表示される「追記型」ではなく、既存の表示内容の影響を受けずに実行する「上書き型」にした。これは、コード全体の機能の流れがわかりやすくなるようにとの配慮からである。受講者に対して「追記型」のコードを複数回実行し演示すると、受講者がどのコードが実行されたのかを把握できなくなる恐れがあると考えたためである。

本ソフトウェアのプレゼンテーションファイルは、HTML 形式であるため、Web サーバにアップロードして外部に公開することが容易である。しかしながら、Web 上で公開することは、上記の設計方針により、セキュリティ上の危険を伴う場合が起こり得る。これを防ぐため、閉じたネットワーク環境で公開する場合を除き、Web での公開時には、サンプルコードを編集できない形に設定することが強く推奨される。

2.7　期待される効果

CodEx のスライドでは、テキストによる説明文、サンプルコード、その実行結果を 1 枚のスライドに統合的に表示できることから、分裂注意効果の影響を軽減できるものと考えられる。さらに、CodEx では講義を実施しながらサンプルコードを編集し、その実行結果を即時に表示することができることから、受講者にフィードバックを与えたり、学習内容の保持と転移を高めるような発問を行うことができると考えられる。

2.8　本章のまとめ

本章では、本章では、Web プログラミング言語教育のためのプレゼンテーションソフトウェア「CodEx」の開発について述べた。CodEx は、スライド上に表示されたコードボックスに HTML/CSS/JavaScript のサンプルコードを示しながら、実行ボックス上にその実行結果を表示し、提示することができるプレゼンテーションソフトウェアである。また、コードボックス上のサンプルコードは、プレゼンテーションをしながらであっても編集することが可能である。また、実行ボックスには、編集後のサンプルコードの実行結果を示すことができる。具体的には、CodEx スライドの例を示しながら、CodEx スライドの構成要素である「コードボックス」と「実行ボックス」の機能と実装について述べた。

第3章

CodExのユーザビリティ評価

本章では、CodEx プレゼンテーションソフトウェアを講義に用いることを想定し、ユーザビリティ評価を行った。

3.1　実験手順

CodEx が前章で定義した要件をみたしているか評価するため、ユーザビリティ評価実験を行った。とくに、受講者としてのユーザビリティと、教師としてのユーザビリティについて評価を行う。

手順は、以下のようになる。

被験者は情報系の研究を行う大学院生・大学教員 10 名である。本実験は、事前アンケート、実験者による模擬授業、ユーザビリティテスト、事後アンケートからなる。この実験の流れを図 3.1 に示す。

実験開始前に、実験者が被験者に対して、CodEx の概要と使い方について説明した。その後、被験者に CodEx を簡単に利用してもらったのちに事前テストに解答してもらった。

実験者による模擬授業では、「CSS のイントロダクション」に関する内容について被験者に受講してもらった。この授業では、CSS の基本的な文法について、スライド上にサンプルコードを示した。その上で、スライド上に表示されたサンプルコードを編集し、関連するサンプルを示しながら授業を進めた。例を挙げれば、CSS の class セレクタを別のセレクタに置き換えるといった例である。さらに、発展的内容であるが、JavaScript を含んだコードにより CSS のセレクタの再現を試みるような内容も含んだ。

授業内容は、コンピュータを専門としない大学学生に対して教えることを想定したものである。この教材を選んだ理由としては、この授業内容では、HTML と CSS の記述が相互に影響を及ぼしながら機能する内容で、これらの情報を統合的に表示する必要があると考えられるため、適切と判断したためである。

被験者の座席にはノート PC が置かれており、授業で用いられたプレゼンテーションと同様のものが授業を受講しながら操作できる。すなわち、受講者は CodEx を用

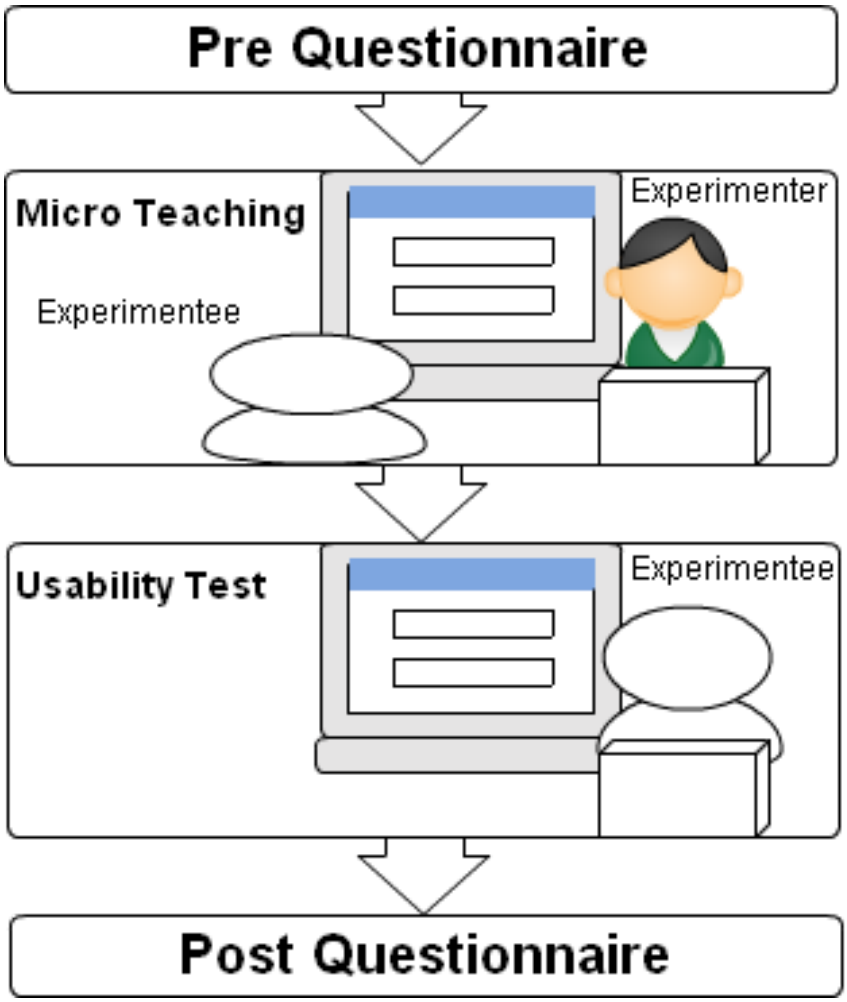
Pre Questionnaire
Micro Teaching
Experimenter
Experimentee
Usability Test
Experimentee
Post Questionnaire

図 3.1　実験 1 の手順

いて、授業と同じ内容のサンプルコードを手元で実行しながら学習することが可能である。

なお、実験時の通信環境による影響をなくすため、受講者・被験者のプレゼンテーションともに、Web サーバーからダウンロードする形式ではなく、手元の PC にローカルファイルとして保存されたものを利用している。

ユーザビリティテストでは、被験者が教壇に立ち、先に行われた模擬授業と同様な授業を繰り返してもらった。

その後、事後アンケートに回答してもらった。事後アンケートでは、7 件法による質問を中心に行った。7 件法によるアンケートでは、平均点に対する中間値検定を行い、有意差がみられる場合にその項目について適切であると判断する。また、コードボックスに、HTML/CSS/JavaScript のそれぞれにおいて、最大でどの程度の長さが必要と思うか尋ねた。

この結果から、定義された要件が適切かどうかを判断する。

3.2　アンケートの結果

事前アンケート

事前アンケートの結果、被験者の平均年齢 26.5 歳（標準偏差 4.55）、男性：女性＝6：4、大学院生 9 名、教員 1 名であった。被験者の HTML/CSS/JavaScript について、いくらかの経験があった。

事後アンケート：7 件法

7 件法による事後アンケート（7：強くそう思う、1：まったくそう思わない）の結果を表 3.1 に示す。表 3.1 は、7 件法によるアンケートの結果の平均点と標準偏差を示している。また、中間値検定（帰無仮説：得られた平均値の平均値は 4 と等しい）による結果を示す。

なお、これらの各設問項目は、以下のような分類を行うことができる。

（1）プレゼンテーションの表示・操作

（2）コードボックス・実行ボックスの表示機能

（3）コードボックスの編集機能

（4）受講者の立場からの評価

（5）教える立場からの評価

（6）外部コンテンツの利用

それらによる分類も表中に示した。

表 3.1　7 件法によるアンケートの結果（7：強くそう思う　1：まったくそう思わない、*：$p < 0.05$　**：$p < 0.01$）

設問	分類	質問文	平均	SD
1	(1)	スクリーンの説明文の文字の大きさは適切だと思いましたか？	5.2**	1.23
2	(2)	スクリーンのコードボックスの文字の大きさは適切だと思いましたか？	5.0*	1.56
3	(2)	スクリーンの実行ボックス内の文字の大きさは適切だと思いましたか？	5.0*	1.49
4	(2)	スライド上にサンプルコードを表示する機能は有用だと思いますか？	6.5**	0.53
5	(2)	説明文・図・サンプルコード・実行結果を同時に表示できる機能は有用だと思いますか？	6.5**	0.85
6	(3)	説明の内容とコードボックス、実行ボックスの内容をうまく関連付けて理解することができましたか？	6.3**	0.67
7	(2)	スライド内でコードを色分けして表示する機能はわかりやすいと思いましたか？	6.0**	0.67
8	(3)	重ね合わせた実行結果を表示する機能は授業を行う上で必要だと思いますか？	6.3**	1.06
9	(3)	サンプルコードを編集できる機能により、生徒の理解が促進されると感じましたか？	6.9**	0.32
10	(3)	サンプルコードを編集できる機能により、受講者と教師間におけるインタラクティブな授業が行えると思いましたか？	5.8**	0.63
11	(2)	「画面を切り替えながら提示する場合」と比較し、本ツールの「画面を切り替えずに提示」する機能は使いやすいですか？	6.5**	0.71
12	(2)	スライド内でサンプルコードを編集し、結果を表示できる機能により、スムーズな授業進行を行うことができると思いましたか？	6.2**	1.23
13	(4)	手元の PC でサンプルコードを実行することは授業内容を理解する上で重要だと思いましたか？	6.6**	0.52
14	(6)	本実験で利用した動画の内容は授業のテーマに即していたと思いますか？	6.1**	0.74
15	(6)	一般的に、授業内で動画を利用することは有効だと思いますか？	5.7**	0.82
16	(5)	本実験で Iframe で表示したページは授業のテーマに即していたと思いますか？	5.9**	0.99
17	(5)	一般的に、授業内に iframe を用いて外部の Web ページを示すことは有効だと思いますか？	4.9*	1.37
18	(1)	プレゼンテーションの操作はスムーズに行えましたか？	5.9**	0.99
19	(2)	スライド内にコードボックス・実行ボックス・説明文を同時に表示できる機能は有用であると思いますか？	6.6**	0.52
20	(3)	本ソフトウェアを利用して授業を受けてみたいと思いましたか？	6.1**	0.74
21	(4)	本ソフトウェアを利用して授業を教えてみたいと思いましたか？	6.0**	1.15

事後アンケート：サンプルコードの長さ

CodEx の要求仕様のひとつとして、1 スライドに表示できるサンプルコードの行数は 20 行としている。

この要求仕様の妥当性を確認するため、コードボックスに表示するサンプルコードの長さについて尋ねた。結果を表 3.2 に示す。

表 **3.2**　コードボックスに必要と思われる最大行数

	行数（回答の平均）	標準偏差
HTML	15.0	7.36
CSS	14.4	8.11
JavaScript	15.4	7.04

3.3　考察

事後アンケート：7 件法

7 件法アンケートの結果、各項目において有意に高い点数が得られた。

分類（1）の結果より、プレゼンテーションの表示・操作が問題なく行えると判断した。

分類（2）の結果より、コードボックス・実行ボックスの表示機能については、適切に表示できていると判断した。

分類（3）の結果より、コードボックスの編集機能については必要であると判断した。

分類（4）の結果より、受講者としての立場から、CodEx が有効と判断した。ただし、本実験では、受講者の学習効果には踏み込んでいない。学習効果に関する実験は、第 6 章で述べる。

分類（5）の結果より、教える立場から、CodEx が有効と判断した。

分類（6）の結果より、外部コンテンツの利用が有効と判断した。

事後アンケート：サンプルコードの長さ

CodEx の仕様では、スライド上に同時に表示できるサンプルコードの行数として 20 行を設定した。この数字は、CodEx が大学の一般情報処理教育向けであることや、HTML のタグを 10 個程度表示したり、JavaScript の構文を 6 個程度組み合わせることができること、スライドの表示面積と文字サイズの兼ね合いから設定した。

アンケートの結果、平均行数は各言語（HTML/CSS/JavaScript）でいずれも 15 行前後におさまっている。そのため、平均との比較では、20 行表示可能という仕様は若干大きいという結果になっている。ただし、この平均行数と 20 行表示可能であることに、有意差はみられなかった。

参考データとして、筆者が過去に大学で行った授業（半期・30 コマ）における、サンプルコードの平均長さと最大長さを表 3.3 に示す。表から、サンプルコードの最大行数は 12 行であった。この数字は、上記のアンケートによる数字と近い。

ところで、最大表示行数には若干の余裕が必要であると考えられる。これは、サンプルコードが長くなった場合に対応しやすいことに加え、スライド上に説明文などが表示された場合には、表示可能なサンプルコードの長さが 20 行よりも小さくなるためである。さらに、授業時の様々なコンテクストに対応するため、また、コードボックスの文字の大きさについて、7 件法アンケートの結果から十分であると判断される。これらのことから、サンプルコードの表示可能な長さが 20 行であることは、妥当であると結論づけた。

表 3.3　筆者が担当した授業科目におけるサンプルコードの行数　極端に長いサンプルコード（HTML ファイル全体をコードボックスに表示したものなど）は除いた

言語	平均	最大値
HTML	3.5	10
CSS	4.0	10
JavaScript	6.7	12
Text	1.0	1
全体	4.9	12

考察のまとめ

実験結果による要求仕様項目の評価をまとめた、総合評価を表 3.4 に示す。この表の項目は、前章で述べた要求仕様に対応し、アンケートによる評価をまとめている。

表 3.4 から、前章で挙げた要求仕様の大半については満たしていると判断される。本実験で評価を行っていない、コードボックスと実行ボックスの対応付けについては、次章以降で説明する「CodEx GUI Editor」による評価を行う。

表 3.4　要求仕様項目の総合評価

要求仕様項目	評価
スライド資料の表示（プレゼンテーションソフトウェアの前提）	○
コードボックスにサンプルコードを表示できる	○
サンプルコードの表示行数は最大 20 行とする	○
コードボックスを逐次編集できる	○
実行ボックスの表示	○
実行ボックスにはラベル名をつけることができる	―
コードボックスには、言語の種類を設定できる	○
コードボックスには、実行先の実行ボックスのラベル名を設定できる	―
実行ボックスにサンプルコードの実行結果を表示できる	○
実行ボックスには、複数のサンプルコードの実行結果を重ね合わせたものが表示できる	○

○：利用可能　―：評価していない

3.4　本章のまとめ

本章では、前章にて開発したCodExについて、前章で定義した要求仕様をみたしているか評価するため、ユーザビリティ評価実験を行った。とくに、受講者としてのユーザビリティと、教師としてのユーザビリティについて評価を行った。

その結果、要求仕様に挙げられていた多くの項目について、要件をみたしていることが確認できた。

本実験では、CodExを用いた講義における学習効果については検討していない。これについては、第6章で述べる。

第4章

スライド作成・編集ソフトウェア「CodEx GUI Editor」の開発

本章および次章では、CodEx によるプレゼンテーションを作成・編集可能にするためのソフトウェア「CodEx GUI Editor」について述べる。

本章では、CodEx GUI Editor の開発について述べる。次章では、CodEx GUI Editor の評価について述べる。

4.1　開発の目的

前章では、CodEx によるプレゼンテーション表示と、その利用について述べたが、プレゼンテーション作成については考慮してこなかった。しかし、実際に授業場面に導入するには、プレゼンテーション作成が十分に容易でなければならない。

CodEx のためのプレゼンテーションを作成するには、HTML ソースを直接編集するか、Wiki のような簡易記法により記述し、コンバータにより HTML へ変換する必要がある。どちらもテキストベースでの編集となり、次章で述べるようにスライドのレイアウトを行う上で不利である。すなわち、図 4.1 に示すように、複数のコードボックスや実行ボックスなどが 1 枚のスライドにおさまるようにする場合、テキストベースでの編集は困難が予想される。これを解決するために、グラフィカルな操作によるスライド編集機能が必要とされる。

本章では、Web プログラミング言語の授業で用いられるプレゼンテーションの作成を容易に・グラフィカルに作成することを可能とするため、プレゼンテーション作成・編集ソフトウェア「CodEx GUI Editor」（以下、Editor と略す）を開発した。

4.2　要求仕様

Editor は、教師のスライド作成を支援することにより、各原理に即したスライド・プレゼンテーション作成を行えるようにするものである。Editor の要求仕様と CodEx の要求仕様、インストラクショナルデザインや Worked Example の原理との対応と

JavaScriptによるスタイルの操作

```
<ul>
 <li class="male" id="father">秀吉</li>
 <li class="female" id="mother">淀殿</li>
 <li class="male" id="son1">鶴松</li>
 <li class="male" id="son2">秀頼</li>
<ul>
```

編集 実行

```
#son1{
        font-style:italic;
}
#son2{
        text-decoration:underline;
}
```

編集 実行

```
var lists = document.getElementsByTagName("li");
for(var i = 0; i < lists.length; i++){
  if(lists[i].className == "male"){
    lists[i].style.fontSize = "40px";
  }

}
```

編集 実行

- 秀吉
  - 淀殿
- 鶴松
- 秀頼

help? contents? restart?　CodEx (hideka 2011)　focus:OFF　sequence:OFF　slide 5/5

図 4.1　複数のコードボックスを表示した CodEx スライドの例

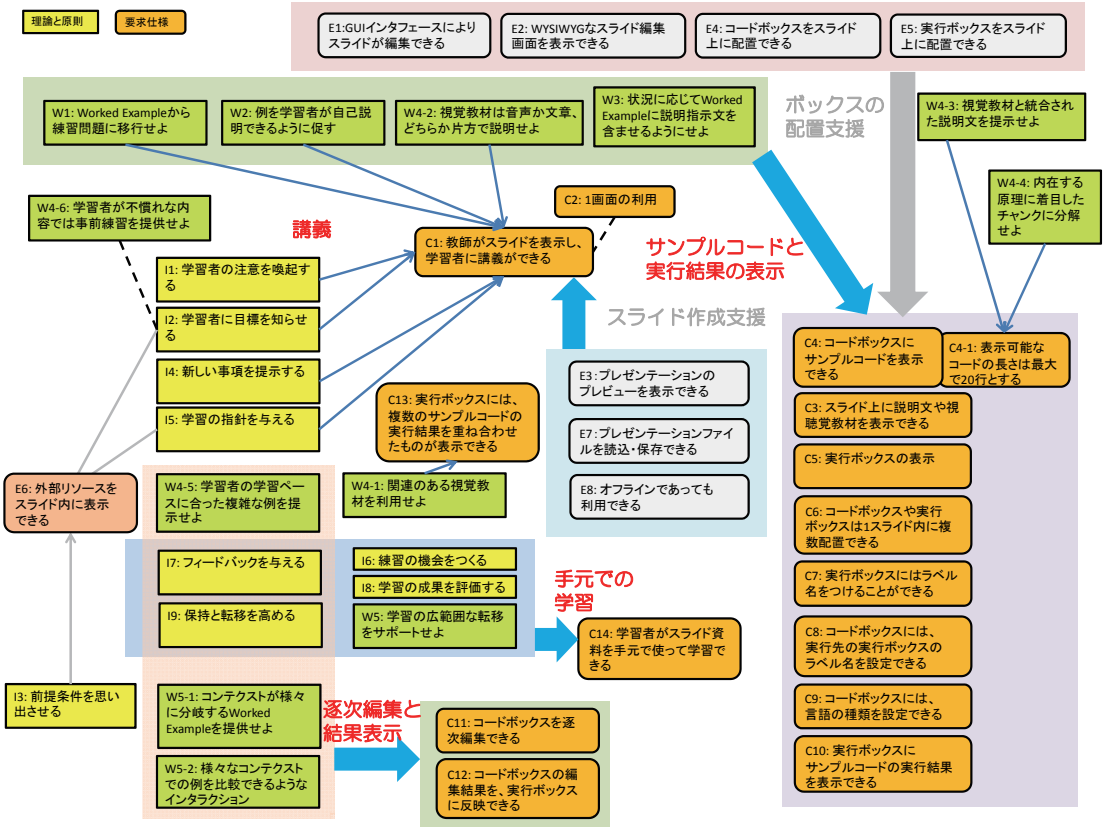

図 4.2　Editor の要求仕様の位置づけ

位置づけを、図 4.2 に示す。図中の着色された部分が、本章・次章で述べる範囲である。

以下、各要求仕様について説明する。

4.2.1　E1: GUI インタフェースによりスライドが編集できる

GUI インタフェースによりスライド編集が可能であること。画面上部の操作パネルに各種操作のためのボタンをまとめて配置する。

4.2.2　E2: WYSIWYG なスライド編集画面を表示できる

スライド画面に表示される各要素（テキスト、図表、コードボックス、実行ボックスなど）をグラフィカルで直観的に（WYSIWYG：What You See Is What You Get）配置できるような機能が利用できること。これにより、複数のコードボックスを 1 枚のスライド上に表示させる場合などのように、スライドレイアウトの自由度を高めることができる。

4.2.3 E3: プレゼンテーションのプレビューを表示できる

スライドをサムネイル表示し、プレゼンテーションの順番上近接するスライドの様子を確認できること。また、必要に応じてサムネイルをクリックすると、そのスライドを編集することができるようになること。

4.2.4 E4: コードボックスをスライド上に配置できる

コードボックスをスライド上に配置できること。配置したコードボックスの位置や大きさをドラッグ操作により変更できること。コードボックス内のサンプルコードをプレゼンテーション中に編集可能であること。Web プログラミングに対応するために、複数のボックスを 1 つのスライドに配置できること。

4.2.5 E5: 実行ボックスをスライド上に配置できる

コードボックス同様に、スライド上に配置・大きさ変更できること。さらに、複数に分かれたサンプルコードに対して、その実行結果を表示するための実行ボックスを指定することができることが必要となる。

E1、E2、E4、E5 の仕様は、「W4-3: 視覚教材と統合された説明文を提示せよ」をより容易にする上で重要と考えられる。

4.2.6 E6: 外部リソースをスライド内に表示できる

Web 上の教材として、リファレンスサイトなどの Web ページや、YouTube の動画などを利用できること。特に、URL の入力とスライド内での配置のためのインタフェースを用意する。

この仕様は、「I2: 学習者に目標を知らせる」「I3: 前提条件を思い出させる」「I5: 学習の指針を与える」上で利用可能と考えられる。

4.2.7 E7: プレゼンテーションファイルを読込・保存できる

既存の CodEx プレゼンテーションファイルを読み込み、編集できるようにする。また、作成したプレゼンテーションをファイルに保存し、発表・講義に利用できるようにする。この機能は、ブラウザで動作するソフトウェアを設計する際に考慮しなければならない問題点である。

4.2.8 E8: オフラインであっても利用できる

オフラインでプレゼンテーション作成・編集が可能である。ただし、E6 に挙げた機能を利用する場合には、オンラインである必要がある。

4.3　既存ソフトウェアとの比較

Editor の実装には HTML5/CSS3 で策定された（策定途中のものも含む）技術を用いている。作成されたスライド・Editor ともにローカルサイドの HTML のみで構成する。すなわち、Editor はブラウザのみで動作し、他のプラグイン等は必要としない。また、インターネット上の Web ページや動画を利用する場合を除き、ネットワーク接続も必要としない。

既存のソフトウェアとの比較を表 4.1 に示す。Amaya Browser/Editor は、GUI による WYSIWYG な HTML エディタである [45]。Amaya を用いればローカル PC 上で Slidy のスライドを作成することができるが [8]、プレゼンテーション作成に特化したツールではない。そのため、プレゼンテーションのプレビューなどを参照できない・編集時のスライド表示が必ずしもプレゼンテーション時のスライド表示と同じでない（スライド編集に関しては完全な WYSIWYG といえない）という問題点がある。

表 4.1　ソフトウェアの機能比較

	Editor	Amaya	Google ドライブ
GUI インタフェース	○	○	○
プレゼンテーションの サムネイルプレビュー	○	○	○
WYSIWYG 編集	○	×	○
外部リソースの表示	○	○	△
サンプルコードの表示領域 （コードボックス）を スライド上に追加・配置	○	△	△
サンプルコードの実行結果 （実行ボックス）を表示できる 領域を追加・配置	○	×	×
コードボックスと 実行ボックスの対応付け	○	×	×
ファイルの保存	○	○	○
オフラインでの使用	○	○	×

○：利用可能
△：一部機能のみ利用可能
×：利用不可能

4.4 CodEx GUI Editor の機能と実装

Editor は HTML5 や CSS3 における規格策定途中の技術を多数用いた。これらの技術は、ブラウザにより現在までの対応状況が大きく異なる。将来的には多くの技術がほとんどのブラウザで利用可能になるとみられるが、現状で厳密なクロスブラウザ対策を行うのは難しい。本論文では Firefox に絞って開発を行った。

4.4.1 画面レイアウト

Editor の画面レイアウトを図 4.3 に示す。「プレビュー」「操作パネル」「スライド編集・表示」の 3 部分から構成される。以下、それぞれの部分について説明する。

サムネイルプレビュー（左側）

スライドの流れがわかるように、プレゼンテーションスライドのサムネイルが表示される部分である。プレビュー部のサムネイルは、現在編集しているスライドの内容が逐次反映される。プレビューをクリックすると、そのスライドが「スライド編集・表示」部に表示され、編集が可能になる。

本機能は、CSS3 で実装された「背景の element 指定」を利用して実現している。CSS3 では、element 指定を行うことで、HTML ドキュメントに表示されている要素を選択し、背景画像のように利用できる。この表示は、実際の HTML での表示に対してリアルタイムに反映される。この機能を利用して、各スライドのサムネイルを作成している。ただし、この機能は実際に表示されている HTML 要素でしか利用できないという問題点がある。そのため、スライド編集・表示部分で表示されているスライド以外も HTML 画面のどこかで表示されている必要がある。 Editor では、CSS の z-index プロパティを用いて、スライド編集・表示部分の隠れている部分に要素をコピーし表示させることで、この問題を解決している。

操作パネル（上部）（GUI インタフェース）

スライドに様々な要素を追加したり、要素の属性（表示の大きさ、文字の色など）を設定したりする際に用いる。アイコンをクリックすると、各機能の詳細なパネルが表示される。各機能については後述する。

スライド表示・編集（WYSIWYG 編集）

ここには、それぞれのスライドが表示される。スライドの構成要素をクリックすると、クリックした要素のテキストを編集したり、移動・大きさ等を変更したりすることができる。図 4.3 で作成しているスライドは、HTML 文書と JavaScript のサンプルコードを示し、それらを表示・実行することができる。実装には、編集対象となるスライドを iframe で読み込み、HTML5 の contenteditable 属性を利用し、閲

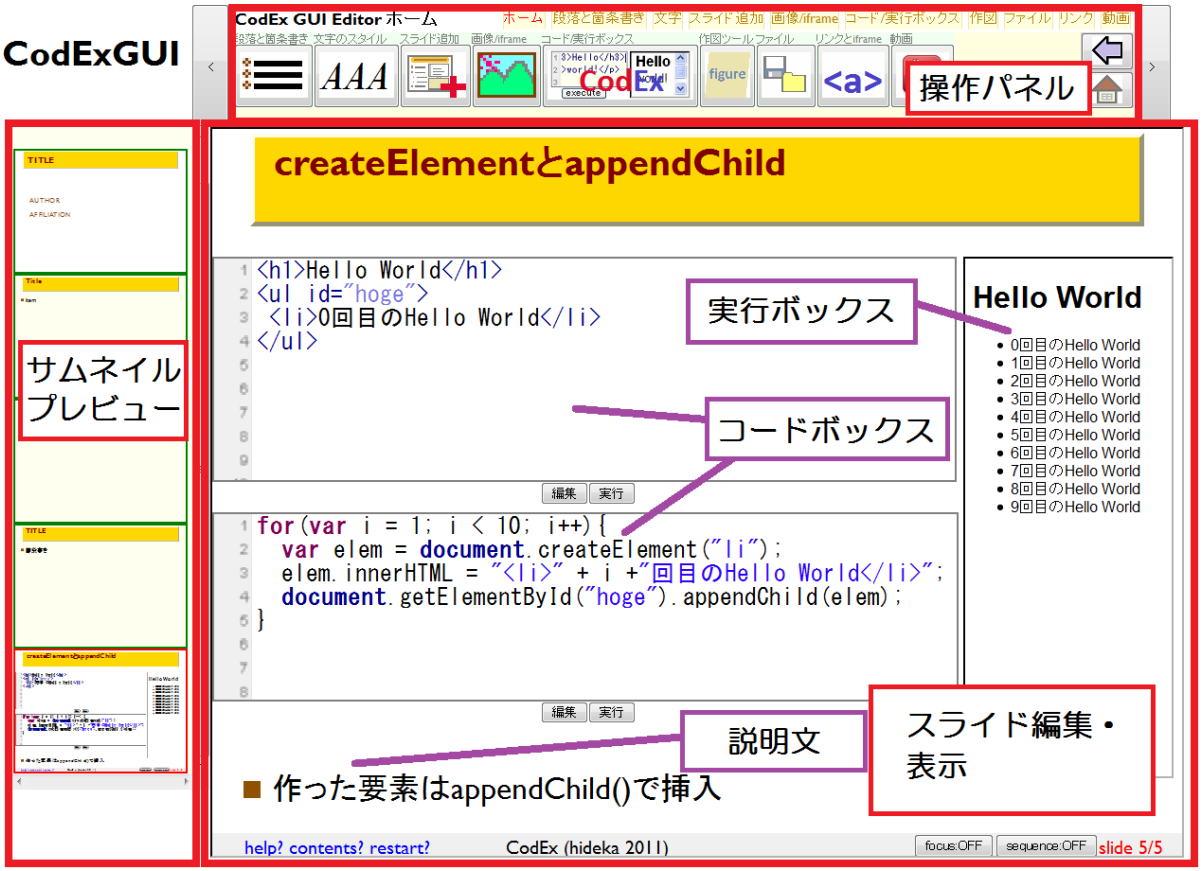

図 4.3　Editor の画面レイアウト

覧者に編集許可を与えることを実現している。ただし、この機能は、既存の HTML 要素内に含まれる文字を編集するなど、ごく簡単な機能が提供されるにとどまる。本論文では、以下に述べるような操作パネルの機能や画像の編集機能を実装した。

4.4.2 Editor 機能の詳細

本節では、Editor の詳細な機能について説明する。

文章と箇条書きの挿入

スライド内に文章や箇条書きなどのテキスト情報を記述するための要素を追加・調整するための機能である。これらの機能は contenteditable 標準の機能だけでは実現できないことに注意する必要がある。contenteditable 属性は、HTML の論理構造（DOM）に修正を加えることを意図していないからである。そのため、本論文では HTML の論理構造（DOM）を制御しながら編集を行うための機能を JavaScript を用いて提供した。具体的には、操作パネルやキーボードの操作等に合わせて要素の追加・削除を行うなどを行っている。

文章の挿入 スライド内に文章を挿入することができる。HTML の p 要素に相当する。

新しい箇条書きの挿入 新しい箇条書きを追加する。HTML の ul 要素に相当する。箇条書き内の項目を追加する際には、箇条書き項目上で Enter キーを押す。

箇条書きの段落下げ 箇条書き項目の段落を下げて記述することができる。Tab キーを押すことで、段落が下がる。

箇条書きの段落上げ すでに下がっている箇条書きの項目の段を上げる場合は、Shift + Tab キーを押すことで、段落が上がる。

文字の装飾の変更

スライド内に文章や箇条書きなどのテキスト情報を記述するための要素を追加・調整することができる。

スライドの追加

スライドを追加することができる。スライドにはいくつかのテンプレートが用意されており、ユーザはそのテンプレートから選択してスライドを追加する。例えば図 4.3 で編集中のスライドのように、左側に複数のコードボックス、右側に実行ボックスを表示したスライドといった複雑なレイアウトを作る際に有用である。

コードボックス・実行ボックスの追加・配置

コードボックスや実行ボックス（ボックス類）を追加する機能である。スライド内でサンプルコードやその実行結果を表示するには、サンプルコードを表示する「コードボックス」や、その実行結果を表示するための「実行ボックス」をスライド内に配置する必要がある。 ボックス類の追加を行うには、操作パネルのボックス追加ボタンを押す。

配置されたボックス類の位置や大きさを変更するには、変更したいボックスを選択する。選択された要素は、「編集可能状態」となり、色付きのボックスで表示される。編集可能状態に切り替えた場合のコードボックス・実行ボックスの例を図 3 に示す。編集可能状態では、マウス操作により位置やボックスの大きさを変更できる。具体的には、ボックスの右下をドラッグすることで大きさの変更ができる。また、それ以外の部分をドラッグすることで位置の移動ができる。編集可能状態を終了すると、図 4.4 中央のコードボックスのように、通常の表示に戻る。

「編集可能状態」の実装としては、選択されたボックス類を div 要素に置き換えている。この div 要素は、元のボックス類のサイズ等の情報を引き継ぐが、ドラッグ操作により移動やサイズ変更を行うことができる。また、div 要素の子要素として、属性を入力する input 要素などをもち、ここの文字列を編集することでユーザは後述のラベルなどの設定が可能である。この編集可能状態ボックスの「決定」ボタンを押すと、編集可能状態ボックスのサイズ情報等を引き継いだ元のボックス類に置き換えられる。

HTML5 において、ボックス類を構成する iframe 要素をドラッグ操作で変形することは容易でない。本研究では、div 要素を変形させるようにすることで、グラフィカルな配置変更を実現した。

コードボックス・実行ボックスの対応付け

また、編集可能状態となった要素では、実行ボックスにラベルを設定し、コードボックス側で実行先の実行ボックスのラベルを指定することで対応付けることができる。Web プログラミングでは複数言語のコードを重ね合わせた実行結果を得たい場合が多いが、この対応付け手法により複数のコードボックスをひとつの実行ボックスに対応づけることができる。ラベルの対応は同一スライドのボックス同士で自動的に対応付けられるが、ユーザが独自に設定することもできる。特に、図 4.1 の例のように、多数のコードボックス・実行ボックスを含むスライドでは、各要素を編集可能状態にして対応付けを確認しながら作業する必要がある。

ファイルの入出力（保存）

編集したいスライドファイルを読み込んだり、ファイルを保存したりできる、ブラウザベースで動作するソフトウェアの場合、特にファイルの保存方法が問題となる。ファイルの保存には、外部サーバやプラグインを必要としない data URL Scheme[32]

図 4.4　編集可能状態のコードボックス（左下）と実行ボックス（右）（左上は通常表示のコードボックス）

を用いているが、将来的にはHTML5で制定されつつあるFile APIのFile Interface[1]を用い、より自由度の高いファイル操作を実装予定である。

外部 Web ページの取り込み

iframeを利用して、外部のWebページをスライド内に挿入することができる。ユーザは、操作パネルのURI入力欄にURIを入力し、挿入ボタンを押すと、編集中のスライド内にiframeが挿入される。

動画の挿入

YouTubeの動画をスライド内に挿入することができる。ユーザは、YouTubeムービーのURIや動画の表示サイズを入力する。挿入ボタンを押すと、編集中のスライド内に動画が挿入される。

その他の機能

その他の機能として、「画像ファイルの挿入」「図形の挿入」「ハイパーリンクの挿入」「スライド内の要素のHTML属性やスタイル属性の直接編集」などがある。

4.5　本章のまとめ

本章では、CodEx GUI Editor（Editor）の開発について述べた。これは、CodExプレゼンテーションの作成を容易に・グラフィカルに作成することを可能とするために開発された。Editorを用いることで、コードボックスや実行ボックスを含むスライドをWYSIWYGに作成することができる。なお、Editorの評価については、次章で述べる。

第5章

「CodEx GUI Editor」のユーザビリティ評価

本章では、CodEx GUI Editor についての評価を行った。

CodEx GUI Editor（Editor）により、教員がスライドを容易に作成可能であるかどうか、ユーザビリティ評価を行った。具体的には、「CSS のセレクタ」について教える授業にて、第 1 章の図 1.2 で説明したステップを授業で行うことを想定する。その上で、本ソフトウェアによるスライド作成を行ってもらい、本研究で実装したソフトウェアの機能が有効であるかを確認することが目的である。

5.1　実験手順

評価実験では、以下の項目を行った。

- 事前アンケート（被験者属性の確認が目的）
- 実験者（教員役）による模擬授業受講
- 被験者による模擬授業で用いたスライドの作成
- 被験者による模擬授業（作成したスライドを使用）
- 事後アンケート

実験手順を図 5.1 に示す。教員役である実験者は、筆者の 1 人が担当した。被験者は、情報系を専攻する大学生・大学院生・教員 11 名で、平均年齢 26.3 歳であった。1 回の実験ごとに、被験者は 1 人である。

Editor では、HTML/CSS/JavaScript を表示・実行することができるスライドを作成可能であるが、本実験ではスライド作成のユーザビリティ評価が目的であること、スライドの表示内容を確認するには十分であると考えられるため、HTML/CSS を教えるためのスライド作成を題材とした。

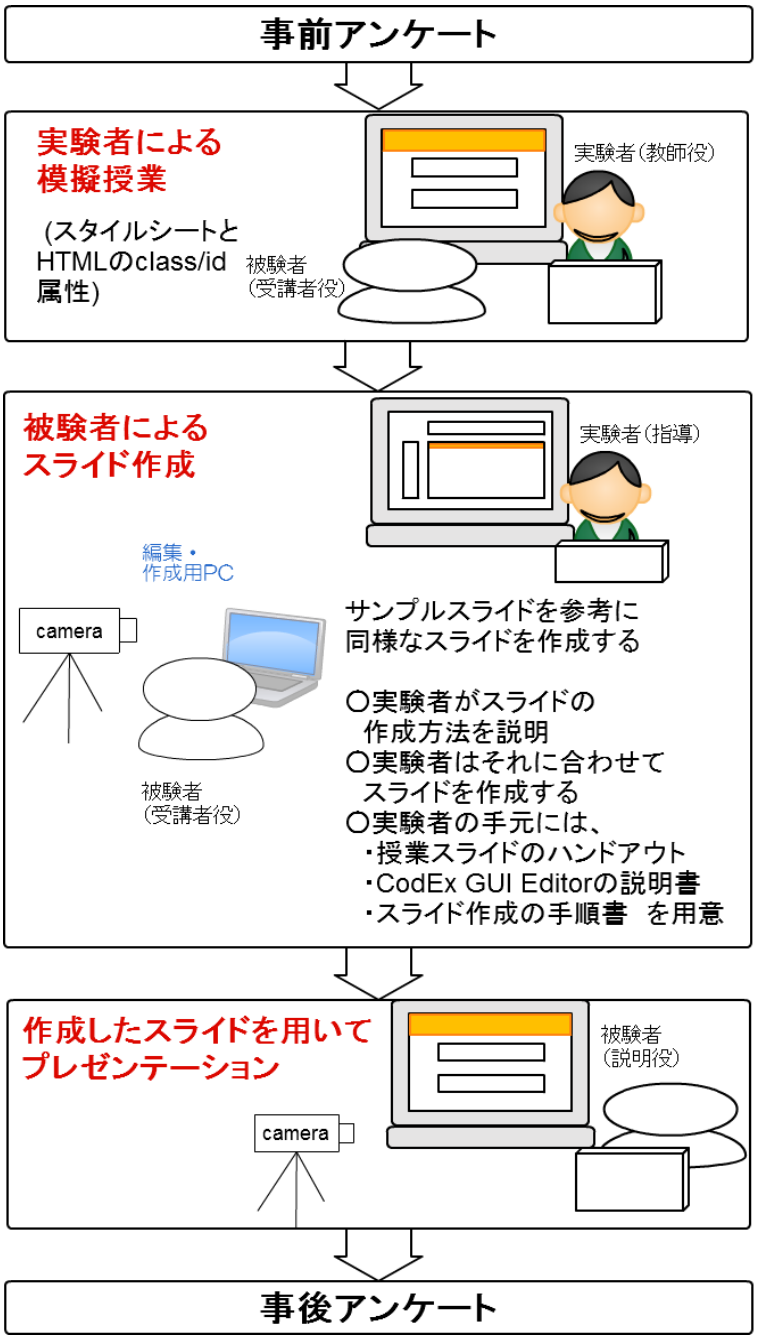

図 5.1　Editor の評価実験手順

実験者による模擬授業では、CSS の「セレクタ」に関する説明講義を、5 分程度行った。講義のプレゼンテーションには、HTML クイックリファレンス [25] の「セレクタの種類」のページを提示したスライド、ドットインストール [9] の動画教材「セレクタを理解する」を提示したスライドを含む。

被験者によるスライド作成時には、実験者が、被験者の作業にあわせて逐次説明を行いながら、スライド作成を行ってもらった。この際、被験者には模擬授業スライドのハンドアウトと Editor の説明書、スライド作成の手順書を配布している。ただし、実験者が逐次説明を行ったため、ハンドアウト以外の資料を閲覧した被験者はいなかった。

作成したスライドは、以下の 6 枚である。

1. タイトル
2. 目次
3. HTML とセレクタの概要（iframe により、Web リファレンスを表示した）
4. セレクタの種類（説明文とビデオ教材を表示）
5. HTML と class、id（左側半分に説明文、右側に HTML のサンプルコードと実行ボックス）
6. セレクタ・class 指定と id 指定の比較（左側に HTML のサンプルコード 1 つと CSS のサンプルコード 2 つ、右側に実行ボックス）

なお、実験内で利用した教材のうち、Web リファレンスとしては HTML クイックリファレンスの記事「CSS の基本　セレクタの種類」を、ビデオ教材としてはドットインストールの「CSS 入門　セレクタを理解する」を、著作者の許諾を得た上で利用した。

スライド (3)(4) で利用するサイト・動画は、実験者があらかじめブラウザにブックマークしてある。被験者は、別ウィンドウでブラウザを起動し、該当ページを開いて URL をコピーする。その後、被験者はスライド編集ウィンドウに戻り、URL を操作パネル内のテキストボックスにコピーし、ボタンを押すと編集可能な iframe がスライド内に生成される。被験者は、この iframe を編集し、適切なサイズ・配置に調整する。

スライド (6) を作成する際には、あらかじめコードボックス 2 つが左半分に、実行ボックス 1 つが右半分に配置されたスライドのテンプレートを用いることができるが、この課題ではさらに左側にコードボックスを 1 つ追加し、3 つのコードボックスを左半分に表示しなければならない。

作成されたスライドにより授業が可能か確かめるために、被験者によるプレゼンテーションでは、被験者が先に作成したスライドを用いて、簡単な説明を行ってもらった。事後アンケートでは、先に実験者が行った模擬授業に関する質問を行い、Editor のユーザビリティ評価を行った。また、実験終了後に、被験者が作成したスライドについて、ツールの機能的な観点から達成度の採点を行った。

5.2　実験結果

5.2.1　事前アンケートによる被験者属性の確認

被験者 11 名を対象に評価実験を行った。ただし、HTML・CSS の記述未経験者 1 名（大学院生）を除いた 10 名を分析対象とする。10 名（男性:女性=5:5、平均年齢（標準偏差）=26.4（5.02））の属性と HTML/CSS の記述に関する経験を表 5.1 に示す。

表 5.1　被験者の属性

	大学生	大学院生	大学教員
被験者人数計	1	7	2
情報系の授業の経験			
経験なし	0	3	0
過去に教えていた	1	3	0
現在教えている	0	1	2
HTML の記述頻度			
ひととおりの知識はあるが、普段から記述することはない	0	3	1
ときどき記述することがある	0	3	1
日常的に記述している	1	1	0
CSS に対する知識			
スタイルの必要がない HTML のみ記述する	0	0	1
テーブルなどで代用	1	1	1
従来の CSS（Level2）を利用	0	3	0
規格策定中の CSS（Level3）を利用	0	3	0

5.2.2　スライド作成に関する 7 件法アンケート

ツール Editor の各機能について評価を行うため、スライド作成に関して、7 件法（7 : 強くそう思う、1 : まったくそう思わない）にてアンケートを行った。評価の分類として、以下の各分類に関する項目を設けた。

（a）プレゼンテーション作成時の全体の操作

（b）外部コンテンツの利用（外部サイトの記事や動画の利用）

（c1）コードボックス・実行ボックスの利用のうち、ボックスの配置に関する項目

（c2）コードボックス・実行ボックスの利用のうち、ボックスの配置以外に関する項目

（d）作成したプレゼンテーションを表示させる際の操作

これらに関する質問項目の評価により、「スライドが容易に作成可能か」を判断する。結果を表 5.2 に示す。また、評定値の平均の検定についての結果（帰無仮説：評定値の平均が 4 に等しい）にて有意差が認められる場合に評価項目が達成できたものと判断する。

分類（a）の各設問では $p < 0.01$ で有意に高い値を示している。

分類（b）については、$p < 0.01$ で有意に高い値を示している。これらの機能についてのユーザビリティは高いと結論付けられる。

分類（c1）のうち、コードボックスや実行ボックスの配置・大きさの変更に関しては、検定の結果、中央値との有意差はみられなかった。

分類（c2）では、各設問とも $p < 0.01$ で有意に高い値を示した。

分類（d）の各設問においても、$p < 0.01$ で有意に高い値を示した。

表 5.2　スライド作成に関するアンケート（** : $p < 0.01$）

分類	設問	平均	標準偏差
a	全体の流れを確認するのに、プレビュー画面は確認しやすかったですか？	6.3**	0.95
a	プレビュー機能は必要だと思いますか？	6.3**	0.67
b	動画の URL は適切に入力できましたか？	6.9**	0.32
b	動画の配置は適切に行えましたか？	5.7**	1.16
b	iframe で表示する Web ページの URL は適切に設定できましたか？	6.7**	0.67
c1	コードボックスの配置（位置の設定）はしやすかったですか？	4.4	1.84
c1	コードボックスの大きさの変更はしやすかったですか？	4.2	1.62
c1	実行ボックスの配置（位置の設定）はしやすかったですか？	4.6	1.71
c1	実行ボックスの大きさの変更はしやすかったですか？	4.3	1.77
c2	コードボックスと実行ボックスの対応づけは適切に行えましたか？	6.7**	0.67
c2	コードボックスと実行ボックスの内容を比較しやすいように配置できると思いますか？	6.6**	0.70
c2	スライドのテンプレートは、コードボックスや実行ボックスを配置する上で有用だと思いますか？	6.1**	0.99
d	プレゼンテーションの表示は制作時に意図したとおりに表示されていましたか？	6.2**	0.79
d	プレゼンテーションの操作はスムーズに行えましたか？	6.2**	1.03

5.2.3　作成されたスライドの評価

作成されたスライドの各項目において、達成度を確認した。表 5.3 に、スライド番号と、項目の達成度を示す。さらに、4.2.2 節にある評価分類との対応も示した。ただし、プレゼンテーションの操作については、被験者が作成したスライドを操作したものを実験者が確認した結果である。また表中に出てくる「配置」は、スライド内での要素の位置を意味する。

スライド (6) は、図 4.1 に示したスライドのように表示領域をいっぱいに使ったスライドとした。このスライドを作成する際には、あらかじめコードボックス 2 つが左半分に、実行ボックス 1 つが右半分に配置されたスライドのテンプレートを用いることができるが、この課題ではさらに左側にコードボックスを 1 つ追加し、3 つのコードボックスを左半分に表示しなければならない。さらに、このスライドでは、コードボックスや実行ボックスを横に並べて配置する必要がある。ボックスの配置・大きさを微調整して適切に配置しなければ、コードボックス内にスクロールバーが

表示され、コードの全体を表示することができない。「コード全体を表示することができたか」は、ツールの操作性を評価する指標と考えられる。そのため、完成したスライドの採点基準のひとつとした。

この結果から、スライド全体の完成は全員が達成し、プレゼンテーションの操作を行うことができたことがわかる。一方で、コードが全て表示されているかについての各項目では、達成度が低く、このことからコードボックスの大きさを最適な大きさに設定できていないことが伺えた。

表 5.3　作成されたスライドの評価（観点と達成度）

分類	スライド番号	採点項目	達成度
a	全体	スライドの完成	100 %
d	全体	プレゼンテーションの操作	100 %
b	3	右 iframe ページ表示	90 %
b	3	iframe の幅が十分か	90 %
b	4	ムービーが正しい配置で挿入できたか	90 %
c2	5	コードボックス内のサンプルコードの内容	100 %
c1	5	コードボックスの配置	100 %
c1	5	HTML コードが全て表示されている [1]	80 %
c1	5	実行ボックスの配置	90 %
c2	5	コードは実行可能か？	100 %
c1	6	HTML コードが全て表示されている [1]	40 %
c1	6	HTML ボックスの配置	100 %
c1	6	中段 CSS コードが全て表示されている [1]	60 %
c1	6	中段 CSS ボックスの配置	100 %
c1	6	下段 CSS コードが全て表示されている [1]	80 %
c1	6	下段 CSS ボックスの配置	100 %

5.2.4　自由記述

自由記述の欄では、「スライドの作成・編集機能についてのよい点や改善点」について尋ねた。

Editor の操作に関する記述のほとんどがボックスの配置・大きさの編集に関するものであった。ボックスの編集に関する記述について、同様な回答をした被験者の人数をまとめたものを表 5.4 に示す。

[1] ボックスの幅・高さが不十分であると、スクロールバーが表示されコードの全体が表示されない。プレゼンテーション時にはコード全体が表示されている状態が望ましいと考えられるため、コード全体が表示されていない場合に達成、そうでない場合に未達成とした。

表 5.4　ボックス配置・大きさ編集に関する自由記述

記述内容	人数
ボックスの編集モードの切り替えに対する不満・要望	7
スライドのテンプレートだけでは不十分なので、ボックスを追加する機能は必要	2

5.2.5　プレゼンテーション作成時間

撮影されたビデオデータから、各被験者がプレゼンテーション作成に要した時間を測定した。ただし、2 名はビデオ撮影できなかったため、欠損である。結果、8 人の被験者の平均で 2236 秒（37 分 16 秒、標準偏差：285 秒）となった。1 枚のスライドあたりでは 373 秒（6 分 13 秒）である。なお、いずれの被験者も、これまでに Editor を操作したことがない。

5.3　考察

5.3.1　スライド作成に関するアンケート

分類（a）では、スライドを作成するのにプレビュー機能が必要であることが示され、さらに本ソフトウェアが提供する機能が利用しやすいことが示された。このことから、本プレビュー機能は、プレゼンテーションを作成する上で有効であるといえる。

分類（b）での評価より、外部コンテンツの利用は容易であるといえる。

分類（c1）については、操作が容易であると判断できない。その原因としては、編集可能状態にしなければ要素の移動等ができないこと、これらのボックスの要素が重なった場合に、要素上でのドラッグのイベントを受け取ることができなくなる場合があることなどが考えられる。後者は、iframe 上では、iframe 内に表示されている Web ページがイベントを受け取ってしまうため、編集中のスライドに対してイベントが送られないという問題点である。スライド編集中には iframe の代わりに代理のボックス要素を表示することでこの現象の解決は可能である一方、編集状態の表示とプレゼンテーション時のスライド表示が異なってしまう問題が発生する。

分類（c2）において、コードボックス・実行ボックスをスライドに加えて利用するための機能については、高い評価を得られた。このことから、CodEx の機能である、「サンプルコードを表示し、実行結果を表示する」機能をもったスライドを作成すること自体は達成できたといえる。

分類（d）において、完成したスライドは制作時に意図したとおりに表示されており、プレゼンテーションの操作もスムーズに行えたと回答した人が多かった。HTML であるため、例えばウィンドウサイズが異なると、文章などの画面レイアウトがまっ

たく違ったものになるということが発生しうる。被験者が本ツールを用いてプレゼンテーションを行った場合もまったくそのような現象が起こらなかったわけではないが、プレゼンテーションを行う上で支障のないレベルであると被験者が考えていたと思われる。

5.3.2 作成されたスライド・プレゼンテーションの評価

Editor を用いることにより、すべての被験者がスライド資料を作成することができた。特に、スライド (6) のように、複数のコードボックス・実行ボックスを横に並べて配置するような場面でも、スライドを作成することができた。これは、GUI による Editor を開発することで改善できた項目の 1 つであると考えられる。

しかしながら、スライド作成時の細かい調整を必要とする作業については、十分なインタフェースが提供できなかった可能性がある。コードボックスや実行ボックスの配置を変更しなければならない課題については、達成率の低い課題が見受けられた。ここでの達成率は、「スライドが完成した」という基準ではなく、「スライドの表示が細かいレベルで問題がない」というように、より厳格に判断している。7 件法アンケートにて示されたように、これらの要素の配置に関する項目は十分な評価を得られなかった。編集インタフェースの改善が必要であると考えられる一方で、実験のために配置に余裕のないサンプルスライドを提示したことも影響したと考えられる。

その一方で、アンケートでは「スライドのテンプレートは、コードボックスや実行ボックスを配置する上で有用である」との結果が得られている。このことから、テンプレートを充実させることで、配置に対する問題の解決策となる可能性が考えられる。ただし、自由記述では、テンプレートの利用だけでなく、ボックスを自由に追加できたほうがよいとの意見もみられた。

今回の実験では、被験者が本ツールの使用について未経験であった。そのため、被験者のスライド作成は、実験者が逐次説明を行ったうえで行われた。実験者による説明を含めたスライドの作成時間は 1 枚あたり 373 秒であったが、被験者が操作に慣れることで短縮が見込める。本実験では、被験者自身が考えたスライドを作成していない。被験者が、経験者の指導のもとでスライドを作成できたことは確認できた。

5.3.3 考察のまとめ

実験結果による機能評価をまとめた総合評価を表 5.5 に示す。この表の項目は、表 4.1 に対応し、7 件法アンケートと作成したスライドの評価をまとめている。コードボックス・実行ボックスをスライド上に配置する機能については利用可能であるが、改善が必要と判断された。よりスライド作成の自由度を高めるために、インタフェースの改善が必要であると考えられる。

表 5.5　実験結果による機能評価のまとめ

	7件法	作成された E3: スライド	総合評価
プレゼンテーションの サムネイルプレビュー	○	—	○
E1、E2: WYSIWYG 編集	○	—	○
E6: 外部リソースの表示	○	—	○
E4、E5: コードボックス・実行ボックスを スライド上に配置	△	△	△
C10: サンプルコードの実行結果を表示	○	○	○
C7、C8: コードボックスと 実行ボックスの対応付け	○	○	○
E7: ファイルの保存	—	○	○
E8: オフラインでの使用	—	—	—

5.4　本章のまとめ

本章では、前章で開発した CodEx GUI Editor についての評価を行った。具体的には、Editor により、教員がスライドを容易に作成可能であるかどうか、ユーザビリティ評価を行った。その結果、評価を行った機能については利用可能であることが示された。一方で、コードボックス・実行ボックスをスライド内に配置する機能については、アンケート結果から改善の必要性が指摘された。

第6章

CodExを用いた講義の評価

本章では、学生側の視点から CodEx を評価するため、模擬授業による CodEx の評価実験を行った。

6.1　実験目的

CodEx を利用することによる効果を測定するため、CodEx を利用した授業（提案手法による授業、以下「提案手法」と記す）と CodEx を利用しない授業（従来手法による授業、以下「従来手法」と記す）の比較を行った。具体的方法としては、小時間の模擬授業による。本実験では、筆者が「JavaScript によるオブジェクト」についての授業を行った。実験開始前の事前アンケートと事前テスト、事後テスト、テスト直後のアンケート、実験終了時アンケートから評価を行った。

6.2　実験手順

図 6.1 に、実験手順のフローチャートを示す。

被験者は、東京工業大学の工学部情報工学科の学部 2 年生（実験当時）である。

被験者は、ランダムに 2 グループに分けられ、グループ 1 かグループ 2 にあらかじめ割り当てられた。ここで、被験者は、事前アンケートを実施する前に分けられ、どちらかのグループの実験に参加するように指示した。後述するように、本実験では JavaScript に関する事前知識がないような学生に対して分析を行っているが、グループ分けした時点では被験者の正確な属性（JavaScript の事前知識の有無）がわからない状態でグループ分けを行うことになった。これは、被験者に実験開始時まで実験目的を伝えないようにし、実験への影響を軽減するためである。さらに、事前アンケートの実施手順に起因して、2 グループの人数に大きな差が生まれている。最終的に、データを取得した対象はグループ 1 に 11 人、グループ 2 に 5 人となった。

すべての被験者は、C 言語に関する事前テストを模擬授業を開始する前に行った。被験者は、学科の授業にて C 言語について既習である。続いて、被験者は模擬授業セクションを受講した。模擬授業にて、被験者は「JavaScript におけるオブジェク

ト」についての授業を受けた。模擬授業セクションは、4 つに分かれており、それぞれの授業時間は 8 分（指導案上）である。グループ 1、グループ 2 それぞれに対して、提案手法による授業と従来手法による授業を交互に行った。提案手法では、教師役である実験者が教壇で説明する際に CodEx を用いた。従来手法では、教壇で説明する際に、プレゼンテーションソフトウェア（CodEx から、コードボックスと実行ボックスを除いたもの）のほかに、サンプルコード表示用のテキストエディタ（TeraPad）、実行結果表示用のブラウザ（Firefox）を利用した。模擬授業の各セクションのテーマとグループごとの提案手法・従来手法の割り当てを表 6.1 に示す。

提案手法では、図 6.2 に示すように、教師は、コードボックス・実行ボックスを用いて 1 つのスクリーンに「説明文」「サンプルコード」「実行結果」を示した。そのため、被験者は、これらを同時に見ることができた。

従来手法では、図 6.3 に示すように、教師は「説明文」「サンプルコード」「実行結果」を「プレゼンテーション」「テキストエディタ」「ブラウザ」の画面を切り替えながら説明を行った。そのため、被験者は、これらを同時に見ることができなかった。

模擬授業の各セクションで行われている授業内容は、提案手法・従来手法とも同一である。また、事後テストの内容も同一である。

各セクションが終了した後、速やかに事後テストを行った。このテストは紙による試験である。試験の内容は、直前の模擬授業で扱った内容に関連したものとなっている。実験の参加者全員が試験に解答したら、速やかに次のセッションの授業へと移った。

表 6.1　各セクションにおける授業内容とグループ分け

授業内容	グループ 1（n=11）	グループ 2（n=5）
JavaScript の変数と関数	提案	従来
オブジェクト、プロパティ、メソッド	従来	提案
変数のコピーと値呼び出し・参照呼び出し	提案	従来
JavaScript におけるコンストラクタ	従来	提案

6.3　事前テスト・講義・事後テストの内容

6.3.1　事前テストの内容

事前テストでは、被験者の C 言語に関する知識について調べた。被験者のカリキュラム上、既習の内容であることから、すべての被験者は C 言語に関する経験を持っていた。事前テストでは、「サンプルコードの実行結果の予想」「printf 文の利用法」「変数の取り扱い」「関数の宣言」「ポインタ変数の取り扱い」について取り扱った。これらの質問は、模擬授業で取り扱う JavaScript の講義内容と関連していると考えられる。

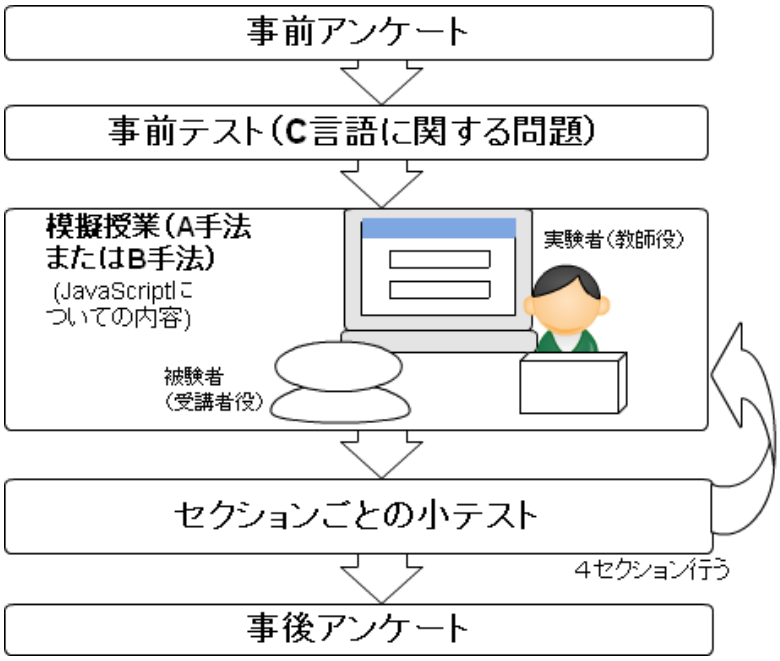
事前アンケート
事前テスト（C言語に関する問題）
模擬授業（A手法
またはB手法）
(JavaScriptに
ついての内容)
実験者（教師役）
被験者
（受講者役）
セクションごとの小テスト
4セクション行う
事後アンケート

図 6.1　実験の手順

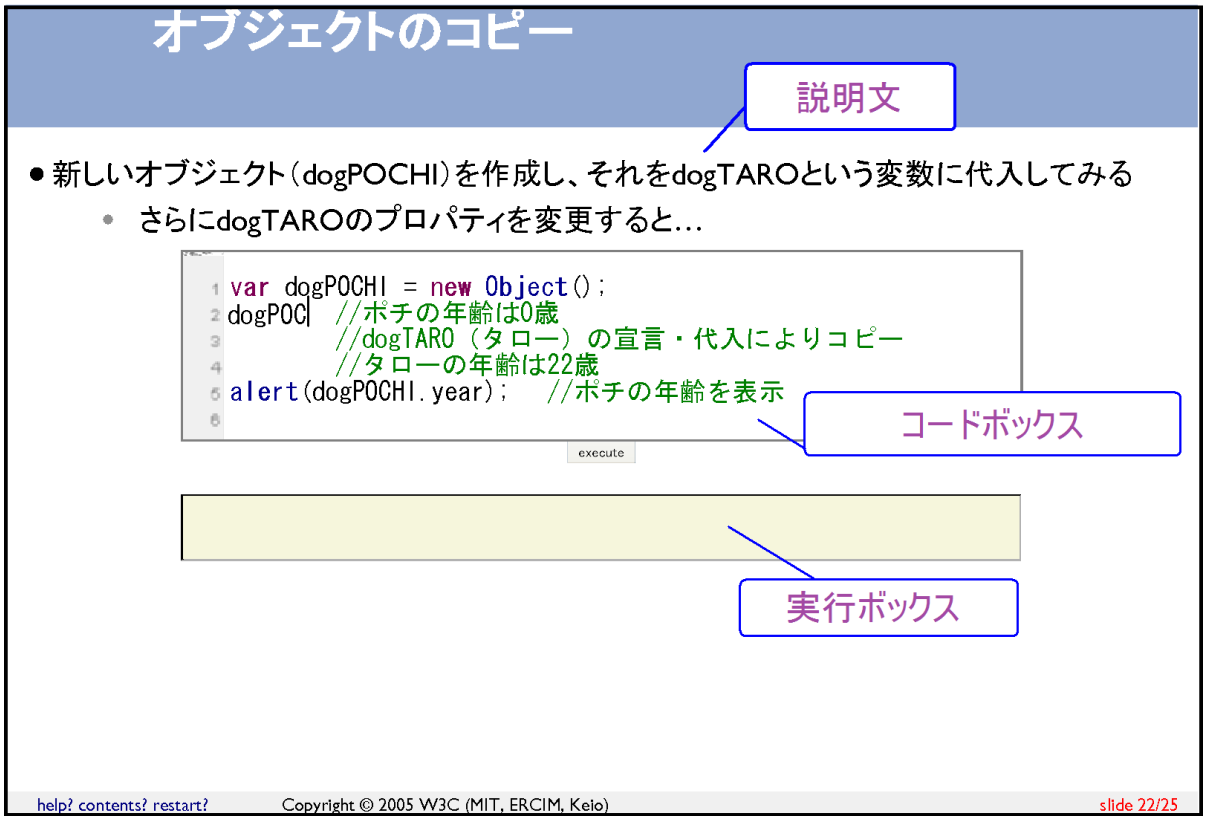

図 6.2　提案手法で用いたスライドの例

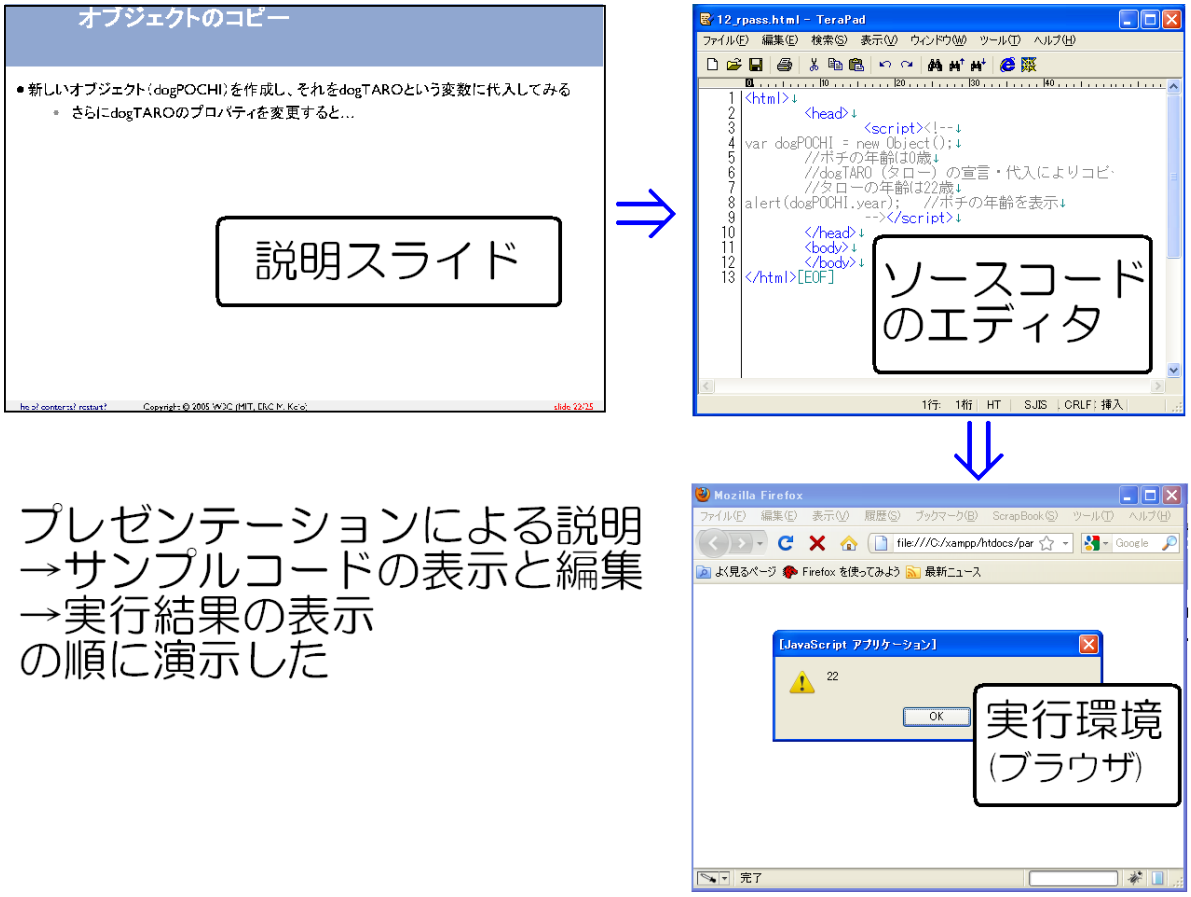

図 6.3　従来手法による授業（画面切り替えを行っていることに注意）

6.3.2　講義内容

講義内容は「JavaScript におけるオブジェクト」である。この講義は、表 6.2 に示すような 4 セクションに分かれている。また、セクション 3 とセクション 4 では、複雑なプログラムの動作メカニズムを説明するために、図 6.4 に示すような形式でサンプルコードを示した。すなわち、最初に未完成のサンプルコードを示し、それを説明しながら編集することで講義を進めた。

各グループの実験の統制のため、被験者からの質問は受け付けていない。また、表に、各セクションで用いたスライド枚数、サンプルコードの数、前述のように編集しながら説明したサンプルコードの数と各セクションごとの講義時間（指導案上の講義時間：各 8 分と実際に講義に要した時間）を示す。

講義中の被験者に、PC は与えられていない。また、講義中にメモをとることも認めていない。

6.3.3　事後テストの内容

各セクションが終了するごとに、被験者は紙による事後試験を受けた。各試験は、「空欄補充」「コード読解」「コード記述」の 3 種類各 5 問からなる。

空欄補充では、説明した内容についての文章について、一部を空欄にし、語句を解答させた。

コード読解では、与えられたコードを実行した結果についての質問や、与えられたコードにおける間違いを指摘させ、どのように修正すべきかを解答させた。

コード記述では、条件を与え、その条件に従って動作する JavaScript のコードを記述させた。

それぞれの問題ごとに 1 点とし、各事後テストの合計は 15 点である。これを。各セクションごとに計 4 回行った。各試験には時間制限を設けていないが、被験者にはできるだけ早く・正確に解答するよう指示した。また、被験者には、一度解答した問題に戻ることができないよう、テスト用紙上の問題配置に配慮した上で、前の問題に振り返って解答しないように指示した。さらに、問題用紙をめくった時刻を記録し、そこから各セクション・各種類の問題の解答に要した時間を算出した。

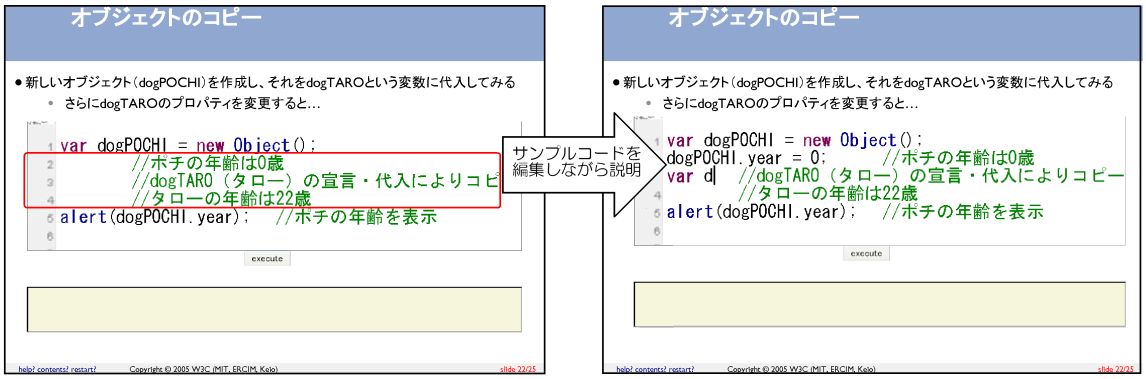

図 6.4　未完成のサンプルコードを完成させながら説明する講義

表 6.2　講義と事後テストの内容

セクション	講義内容	事後テストの内容（各 5 点満点）
1	JavaScript における変数と代入 2 通りの関数の宣言法	空所補充 与えられたコードの修正 関数の宣言に関するコード記述
2	オブジェクトの生成法と プロパティ・メソッド	空所補充 与えられたコードの修正 オブジェクト宣言に関するコード記述
3	プリミティブ型の変数と オブジェクト変数のコピー	空所補充 与えられたコードの実行結果 変数のコピーに関するコード記述
4	JavaScript における コンストラクタの利用法	空所補充 与えられたコードの実行結果 コンストラクタに関するコード記述

6.4　結果

表 6.3　講義時間・スライド枚数とサンプルコードの数

セクション	講義時間			スライドの枚数	サンプルコードの数	左のうち編集したサンプルコードの数
	指導案上の時間	提案手法時の実時間	従来手法時の実時間			
1	8min	7min51s	7min3s	6	6	0
2	8min	8min19s	8min52s	7	4	0
3	8min	7min41s	7min28s	4	3	3
4	8min	7min55s	7min31s	4	3	3

6.4.1　被験者

被験者は 21 人の情報工学科学部生（全員男性）であった。彼らは、カリキュラム上、C 言語について既習であるが、JavaScript について学ぶ機会はない。本実験では、実験開始後に JavaScript の経験があるかどうかについて尋ねたため、実験参加者には JavaScript の未経験者・経験者が含まれている。今回の実験では、実験条件を整えるために、分析対象は JavaScript について未経験の 16 名のみとした。

6.4.2　事前アンケートと事前テスト

事前アンケートでは、被験者にプログラミングの経験について尋ねた。この結果により、分析対象の JavaScript について未経験の被験者を抽出した。

被験者は、グループ 1 とグループ 2 にランダムに割り当てられた。彼らのプログラミング歴はそれぞれ 2.41 年（標準偏差 : 1.69）と 2.80 年（標準偏差 : 2.16）であった。両者に有意差は認められなかった（$F(12, 5) = 0.47$, $p = 0.70$）。

事前テスト（5 点満点）では、それぞれの被験者の C 言語に対する理解を確認した。この内容は、後の JavaScript についての内容と関連するようにしている。多くの被験者で、ポインタに関する問題に間違いが見受けられたが、これは JavaScript の参照渡しによるデータのコピーと類似する概念である。この問題は、ポインタによる理解が十分でないと解答できないように考慮されている。この問題については、グループ 1 で平均点 3.90（標準偏差 1.14）、グループ 2 で平均点 4.00（標準偏差 : 0.71）であった。両者に有意差は認められなかった（$F(11, 5) = 0.37$, $p = 0.87$）。

6.4.3　事後テストの解答時間

事後テスト：セクション 1

セクション 1 終了後の事後テストの解答時間を表 6.4 の上から 1 段目に示す。

空欄補充問題では、提案手法による群が、実験手法による群より短い解答時間となった。また、$0.05 \leq p < 0.10$ で有意傾向がみられた。効果量（Cohen の d）は Large（$0.8 \leq d$）であった。なお、本論文では、効果量の判断は文献 [27] に従った。

コード読解問題では、提案手法による群が、従来手法による群より短い解答時間となった。ただし、両者に有意差は認められなかった。効果量は small（$0.2 \leq d < 0.5$）であった。

コード記述問題では、提案手法による群が、従来手法による群より短い解答時間となった。また、$p < 0.05$ で有意差が認められた。効果量は large（$0.8 \leq d$）であった。

事後テスト：セクション 2

セクション 2 終了後の事後テストの解答時間を表 6.4 の上から 2 段目に示す。

空欄補充問題では、従来手法による群が、提案手法による群より短い解答時間となった。しかしながら、両者に有意差は認められなかった。効果量は、Large（$0.8 \leq d$）であった。

コード読解問題では、従来手法による群が、提案手法による群より短い解答時間となった。しかしながら、両者に有意差は認められなかった。効果量は very small（$d < 0.2$）であった。

コード記述問題では、従来手法による群が、提案手法による群より短い解答時間となった。ただし、両者に有意差は認められなかった。効果量は small（$0.2 \leq d < 0.5$）であった。

事後テスト：セクション 3

セクション 3 終了後の事後テストの解答時間を表 6.4 の上から 3 段目に示す。

空欄補充問題では、　提案手法による群が、従来手法による群より短い解答時間となった。しかしながら、両者に有意差は認められなかった。効果量（Cohen の d）は very small（$d < 0.2$）であった。

コード読解問題では、従来手法による群が、提案手法による群より短い解答時間となった。しかしながら、両者に有意差は認められなかった。効果量は small（$0.2 \leq d < 0.5$）であった。

コード記述問題では、従来手法による群が、提案手法による群より短い解答時間となった。ただし、両者に有意差は認められなかった。効果量は small（$0.2 \leq d < 0.5$）であった。

事後テスト：セクション 4

セクション 4 終了後の事後テストの解答時間を表 6.4 の上から 4 段目に示す。

空欄補充問題では、従来による群が、提案手法による群より短い解答時間となった。しかしながら、両者に有意差は認められなかった。効果量（Cohen の d）は small ($0.2 \leq d < 0.5$)　であった。

コード読解問題では、提案手法による群が、従来手法による群より短い解答時間となった。また、$p < 0.05$ で有意差が認められた。効果量は large（$0.8 \leq d$）であった。

コード記述問題では、従来手法による群が、提案手法による群より短い解答時間となった。また、$0.05 \leq p < 0.10$ で有意傾向がみられた。効果量は large（$0.8 \leq d$）であった。

表 6.4　事後テストの解答時間

セクション 1

	空所補充		コード読解		コード記述	
	提案	従来	提案	従来	提案	従来
平均（秒）	41	49	87	101	261	367
標準偏差	8.50	6.46	24.49	40.59	81.13	58.82
p 値	0.073+		0.500		0.014*	
d	-0.903		-0.476		-1.190	

セクション 2

	空所補充		コード読解		コード記述	
	提案	従来	提案	従来	提案	従来
平均（秒）	64	52	182	174	224	223
標準偏差	12.72	14.66	53.19	39.87	55.35	39.77
p 値	0.13		0.77		0.96	
d	0.81		0.19		0.03	

セクション 3

	空所補充		コード読解		コード記述	
	提案	従来	提案	従来	提案	従来
平均（秒）	63	66	112	103	199	153
標準偏差	19.49	7.60	38.80	11.48	112.66	12.92
p 値	0.69		0.48		0.20	
d	-0.17		0.28		0.49	

セクション 4

	空所補充		コード読解		コード記述	
	提案	従来	提案	従来	提案	従来
平均（秒）	94	83	205	277	410	325
標準偏差	42.10	24.78	47.83	73.67	72.06	63.46
p 値	0.63		0.04*		0.06+	
d	0.34		-0.98		1.12	

注：効果量は Cohen の d を用いた。*: $p < 0.05$, +: $0.05 \leq p < 0.10$

6.4.4　事後テストのスコア

事後テスト：セクション 1

セクション 1 終了後の事後テストの点数を表 6.5 の上から 1 段目に示す。

空欄補充問題では、提案手法による群が、従来手法による群より高い点数を示した。しかしながら、両者に有意差は認められなかった。効果量（Cohen の d）は small（$0.2 \leq d < 0.5$）であった。

コード読解問題では、提案手法による群が、従来手法による群より高い点数を示した。ただし、天井効果が起こっているとみられ、両者とも高い点数で推移している。効果量は small（$0.2 \leq d < 0.5$）であった。

コード記述問題では、従来手法による群が、提案手法より高い点数を示した。ただし、両者に有意差は認められなかった。効果量は large（$0.8 \leq d$）であった。

セクションの点数の合計では、提案手法が従来手法より高い点数を示した。また、両側検定における有意傾向（$0.05 \leq p < 0.10$）が見られた。効果量は large（$0.8 \leq d$）であった。

事後テスト：セクション 2

セクション 2 終了後の事後テストの点数を表 6.5 の上から 2 段目に示す。

空欄補充問題では、従来手法による群が、提案手法による群より高い点数を示した。しかしながら、両者に有意差は認められなかった。効果量）は small（$0.2 \leq d < 0.5$）であった。

コード読解問題では、提案手法による群が、従来手法による群より高い点数を示した。しかしながら、両者に有意差は認められなかった。効果量は medium（$0.5 \leq d < 0.8$）であった。

コード記述問題では、従来手法による群が、提案手法より高い点数を示した。また、両側検定における有意傾向（$0.05 \leq p < 0.10$）が見られた。効果量は large（$0.8 \leq d$）であった。

セクションの点数の合計では、従来手法が提案手法より高い点数を示した。また、両側検定における有意傾向（$0.05 \leq p < 0.10$）が見られた。効果量は large（$0.8 \leq d$）であった。

事後テスト：セクション 3

セクション 3 終了後の事後テストの点数を表 6.5 の上から 3 段目に示す。

空欄補充問題では、従来手法による群が、提案手法による群より高い点数を示した。しかしながら、両者に有意差は認められなかった。効果量は medium（$0.2 \leq d < 0.5$）であった。

コード読解問題では、提案手法による群が、従来手法による群より高い点数を示した。しかしながら、両者に有意差は認められなかった。効果量は small（$0.2 \leq d < 0.5$）であった。

コード記述問題では、従来手法による群が、提案手法より高い点数を示した。しかしながら、両者に有意差は認められなかった。効果量はlarge（$0.8 \leq d$）であった。

セクションの点数の合計では、提案手法が従来手法より高い点数を示した。しかしながら、両者に有意差は認められなかった。効果量はlarge（$0.8 \leq d$）であった。

事後テスト：セクション4

セクション4終了後の事後テストの点数を表6.5の上から4段目に示す。

空欄補充問題では、従来手法による群が、提案手法による群より高い点数を示した。しかしながら、両者に有意差は認められなかった。効果量はmedium（$0.2 \leq d < 0.5$）であった。

コード読解問題では、提案手法による群が、従来手法による群より高い点数を示した。しかしながら、両者に有意差は認められなかった。効果量はsmall（$0.2 \leq d < 0.5$）であった。

コード記述問題では、従来手法による群が、提案手法より高い点数を示した。しかしながら、両者に有意差は認められなかった。効果量はmedium（$0.5 \leq< d < 0.8$）であった。

セクションの点数の合計では、従来手法が提案手法より高い点数を示した。しかしながら、両者に有意差は認められなかった。効果量はvery small（$d < 0.2$）であった。

事後テスト：合計

各事後テストのデータを集計したものを表6.5の最下段に示す。また、グラフ化したものを図6.5に示す。

空欄補充問題では、従来手法による群が、提案手法による群より高い点数を示した。しかしながら、両者に有意差は認められなかった。効果量はmedium（$0.2 \leq d < 0.5$）であった。

コード読解問題では、提案手法による群が、従来手法による群より高い点数を示した。また、両側検定における有意差（$p < 0.05$）が見られた。効果量はmedium（$0.5 \leq d < 0.8$）であった。

コード記述問題では、従来手法による群が、提案手法より高い点数を示した。しかしながら、両者に有意差は認められなかった。効果量はsmall（$0.2 \leq d < 0.5$）であった。

セクションの点数の合計では、従来手法が提案手法より高い点数を示した。しかしながら、両者に有意差は認められなかった。効果量はsmall（$0.2 \leq d < 0.5$）であった。

表 6.5 事後テストのスコア

セクション 1								
	空所補充		コード読解		コード記述		合計	
	提案	従来	提案	従来	提案	従来	提案	従来
平均	4.73	4.60	4.91	5.00	1.73	0.60	11.36	10.20
標準偏差	0.65	0.55	0.30	0.00	1.42	0.89	1.29	0.84
t 検定の p 値	0.709		0.519		0.128		0.088+	
効果量	0.21		-0.43		0.95		1.07	
セクション 2								
	空所補充		コード読解		コード記述		合計	
	提案	従来	提案	従来	提案	従来	提案	従来
平均	4.20	4.55	4.80	4.36	2.60	4.27	11.60	13.18
標準偏差	1.30	0.69	0.45	0.81	1.95	1.19	1.82	1.47
t 検定の p 値	0.492		0.283		0.050+		0.084+	
効果量	-0.33		0.67		-1.04		-0.96	
セクション 3								
	空所補充		コード読解		コード記述		合計	
	提案	従来	提案	従来	提案	従来	提案	従来
平均	3.36	4.20	4.45	4.20	3.09	4.20	10.91	12.60
標準偏差	1.80	0.84	0.69	0.84	1.58	0.45	2.63	1.14
t 検定の p 値	0.346		0.530		0.151		0.195	
効果量	-0.59		0.33		-0.96		-0.84	
セクション 4								
	空所補充		コード読解		コード記述		合計	
	提案	従来	提案	従来	提案	従来	提案	従来
平均	3.20	4.09	3.80	3.36	4.20	3.73	11.18	11.20
標準偏差	1.79	1.04	0.84	0.92	1.30	1.27	2.64	3.03
t 検定の p 値	0.225		0.450		0.357		0.901	
効果量	-0.61		0.49		0.25		-0.01	
合計								
	空所補充		コード読解		コード記述		合計	
	提案	従来	提案	従来	提案	従来	提案	従来
平均	3.94	4.34	4.56	4.13	2.72	3.41	11.22	11.88
標準偏差	1.50	0.83	0.67	0.91	1.71	1.72	2.11	2.25
t 検定の p 値	0.186		0.032*		0.114		0.233	
効果量	-0.34		0.55		-0.40		-0.30	

注：効果量は Cohen の d を用いた。*: $p < 0.05$, +: $0.05 \leq p < 0.10$

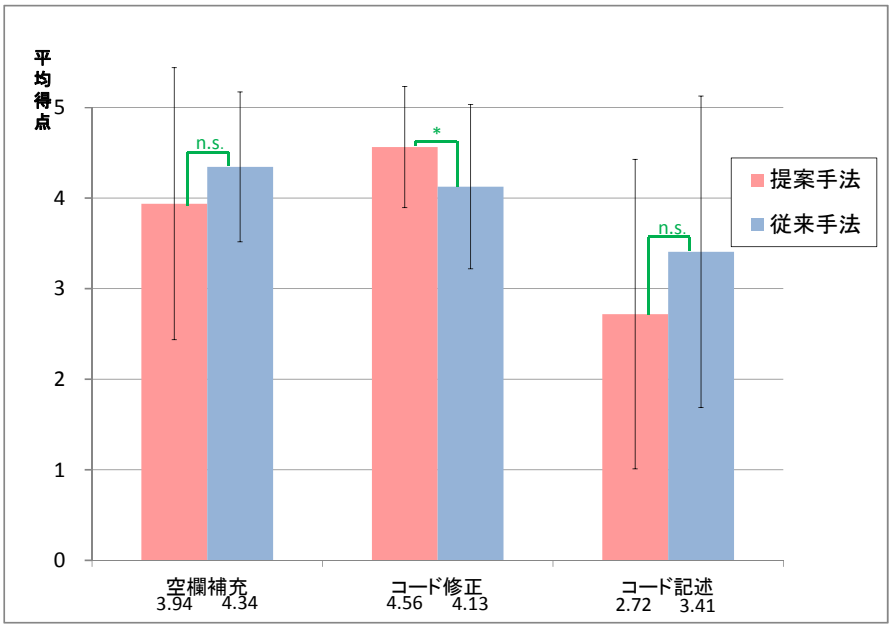

図 6.5　事後テストのスコア（各セクションの平均）

6.4.5　事後アンケート

事後テストの解答後すみやかに、事後アンケートを行った。事後アンケートでは、

- 授業内容の難しさ（5 段階で、5 点が最も難しい）
- 授業の説明のわかりにくさ（5 段階で、5 が最もわかりにくい）

について尋ねた。事後アンケートは、各セクション終了時に行っているため、合計 4 回行っている。

Chang らによれば、授業内容の難易度は内的認知負荷、説明のわかりにくさは外的認知負荷を表す [43]。このような質問による手法は、Gopher・Braune[7] や Paas[18] にあるように、認知負荷や Mental Effort の測定指標として広く利用されている。

これらの結果を、表 6.6 に示す。

事後テスト：セクション 1

授業内容の難しさは、提案手法のほうが従来手法よりも低かった。また、t 検定の結果、$0.05 \leq p < 0.10$ で有意傾向がみられた。効果量は large であった。

授業の説明のわかりにくさは、提案手法のほうが従来手法よりも低かった。また、t 検定の結果、両者に有意差はみられなかった。効果量は very small であった。

事後テスト：セクション 2

授業内容の難しさは、従来手法のほうが提案手法よりも低かった。また、t 検定の結果、両者に有意差はみられなかった。効果量は medium であった。

授業の説明のわかりにくさは、提案手法のほうが従来手法よりも低かった。また、t 検定の結果、両者に有意差はみられなかった。効果量は very small であった。

事後テスト：セクション 3

授業内容の難しさは、従来手法のほうが提案手法よりも低かった。また、t 検定の結果、、$0.05 \leq p < 0.10$ で有意傾向がみられた。効果量は large であった。

授業の説明のわかりにくさは、従来手法のほうが提案手法よりも低かった。また、t 検定の結果、両者に有意差はみられなかった。効果量は medium であった。

事後テスト：セクション 4

授業内容の難しさは、提案手法のほうが従来手法よりも低かった。t 検定の結果、両者に有意差はみられなかった。効果量は small であった。

授業の説明のわかりにくさは、提案手法のほうが従来手法よりも低かった。また、t 検定の結果、両者に有意差はみられなかった。効果量は small であった。

表 6.6　事後アンケートのスコア

セクション 1

質問	内容の難しさ		説明のわかりにくさ	
手法	提案	従来	提案	従来
平均点	1.27	1.80	1.27	1.40
p 値	0.063+		0.716	
d	-1.15		-0.19	

セクション 2

質問	内容の難しさ		説明のわかりにくさ	
手法	提案	従来	提案	従来
平均点	2.20	1.91	1.20	1.27
p 値	0.287		0.818	
d	0.59		-0.11	

セクション 3

質問	内容の難しさ		説明のわかりにくさ	
手法	提案	従来	提案	従来
平均点	2.18	1.60	1.45	1.20
p 値	0.090+		0.343	
d	1.01		0.52	

セクション 4

質問	内容の難しさ		説明のわかりにくさ	
手法	提案	従来	提案	従来
平均点	2.20	2.45	1.40	1.64
p 値	0.395		0.536	
d	-0.44		-0.31	

+：$0.05 < p < 0.10$ 5 件法の平均

・5 が最も難しい、あるいはわかりにくい

6.4.6　実験終了時アンケート

実験終了時アンケートでは、以下の各項目において、提案手法による授業と従来手法による授業のどちらがよいかを 2 択形式で尋ねた。

1. どちらの説明が聞き取りやすかったか
2. どちらの説明が理解しやすかったか
3. どちらの授業のほうが講義内容を覚えられる
4. どちらの授業のほうがプログラミングの授業に適していると思うか

アンケート結果を表 6.7 に示す。問 1、2、4 では、提案手法を選択する学生が多かった。とくに、問 1 では 4 分の 3 の被験者が提案手法を選んだ。一方の問 3 では、両者の回答数が同じとなった。この結果から、CodEx は講義を受講する上で説明を聞き取りやすくする可能性が示唆された。一方で、問 3 からは、CodEx の利用が、学生が授業内容を覚えることには直結しないと考えていることが推測された。

表 6.7 実験終了時アンケートによる回答（提案手法・従来手法のどちらが適しているかを 2 択で回答してもらった）

質問	提案	従来
1. どちらの説明が聞き取りやすかったか	12	4
2. どちらの説明が理解しやすかったか	10	6
3. どちらの授業のほうが講義内容を覚えられるか	8	8
4. どちらの授業のほうがプログラミングの授業に適していると思うか	9	7

6.5 考察

本節では、「事後テストの解答時間」「事後テストのスコア」「事後アンケート」のデータをもとにした考察を述べる。

続いて、Paas・van Merriënboer の図 [19][18] をもとに、「事後テストの解答時間」「事後アンケート」と「事後テストのスコア」から、提案手法および従来手法の教授効率を求める。

6.5.1 問題の解答時間

問題の解答時間は、認知負荷を表す指標のひとつとして用いられ、特に学習者の問題解答時のパフォーマンスを示す指標として用いられる [2][18]。この数値が小さいほど、学習者が授業時間に学習内容を理解し、その結果として認知負荷を軽減したものと考えることができる。

解答時間については、セクション 1 のコード記述・セクション 4 のコード読解で提案手法のほうが有意に短い解答時間となった。これらの項目では、提案手法が、効果的な学習を促しているといえる。一方で、セクション 4 のコード記述では、従来手法のほうが短い解答時間（$p < 0.10$ で有意傾向あり）となるなど、ばらつきもみられる。

6.5.2 分裂集中効果と事後テストの結果

分裂集中効果という概念は、学習における認知負荷理論から派生したものである [39]. これまでの研究では、統合された情報を与えるように設計されたソフトウェア

が、分裂情報効果を軽減するとされてきた。これに従うなら、複数のソフトウェアを切り替えて講義を行うことが、分裂情報効果を与えてしまうと考えられる。

ところで、2 画面を用いて複数のソフトウェア画面を表示した場合に分裂集中効果が起こることが多いと考えられる一方で、2 画面を利用した場合でも分裂集中効果が起こらなかった例が示されている [46][16]。それだけでなく、Chang らや Kuo らによれば、1 画面を利用した場合であっても分裂集中効果が起こりうるとされる。ただし、これらの研究では、1 画面に 2 つのソフトウェアを表示している。これと比較して、本研究では、CodEx という 1 つのソフトウェア上で、「説明」「サンプルコード」「実行結果」という複数の講義上の要素を提示することを試みているという違いがある。

事後テストのコード読解問題においては、天井効果が起こっていたと考えられるセクション 1 をのぞいて、提案手法のほうが従来手法と比較して高い成績を示していた。セクション 1 では、問題の内容が簡単すぎたと考えられる。各セクションごとに提案手法・従来手法の点数における t 検定を行ったが、有意差はみられなかった。一方で、コード読解問題のすべての試験結果において t 検定を行ったところ、5 動作メカニズムの理解について調べた。このことから、より詳細な検討が必要ではあるものの、CodEx による講義は、学生に JavaScript コードの動作メカニズムを教える上で有効と考えられる。この結果を、分裂集中効果理論と照らし合わせてみる。分裂集中効果理論では、複数の情報源からなる情報を統合して理解しようとする際に影響が大きくなる傾向が示されている。特に、表計算ソフトウェアにおける Cerpa らの研究 [35] で、この結果が示されている。学生が、サンプルコードの動作メカニズムを理解する場面では、「説明」「JavaScript の文法」「コード上における動作の表現」「実行結果」といった複数の情報源からなる情報を統合する必要があると考えられる。このことから、本実験で講義を行った内容のうち、情報統合が必要とされる動作メカニズムを理解する場面において、Cerpa らの研究と同様に CodEx による統合された情報を与えたほうが、従来の手法に比べてよい点数を示したものと思われる。

事後テストの空欄補充問題においては、有意差はみられなかったものの、従来手法のほうが高い点数を示す傾向があった。このひとつの原因として、スライド内に複数の情報を同時に表示することによって、認知負荷が上昇したことが原因と考えられる。今後の課題として、ひとつのスライドにあまり多くの情報を載せすぎないようにするべきであるといえる。加えて、この試験問題で尋ねたような、単純な語句の記憶の場面では、情報を統合して表示することに対するメリットが現れない可能性がある。実際、Cerpa らの研究においても、インタラクションの小さい課題、言い換えれば認知負荷の小さいと考えられる課題においては、統合されていない教材・統合されている教材間に有意差がみられていない。ここでは、インタラクションの小さな課題は、容易にワーキング・メモリに情報を保持できるからと説明されている。本実験でも同様の傾向を示したものと思われる。

事後テストのコード記述問題では、セクションごとに異なる結果が得られた。セクション 1 と 4 では、従来手法のほうが高い点数となった一方で、セクション 2 と

3 では逆の結果となった。この理由としては、個人による内容の理解度の差、ひいては点数差が大きくなり、効果量が大きく、p 値が小さくなる結果となったものと考えられる。そのため、この実験結果からどちらの手法が適切であるかといった疑問への結論を得ることは難しい。ところで、コードを記述するという行為は、言語の文法やプログラムの動作メカニズムの理解といった複雑な知識の十分な理解を必要とする。プレゼンテーションソフトウェアによって、これらをすべて教えるのは難しい。これらを習得のためには、学生がコードを実際に記述することが必要と考えられる。実際に、プログラミングの習得には「コードを記述する」ことが最良の学習法だと信じられている。ただし、本論文では、あくまで講義形式によるプログラミング言語教育の導入の補助を想定している。「プログラムコードを書くことができる」という学習目標は、本論文の対象とする範囲の先に位置するものであるといえる。

6.5.3 事後アンケートの結果

学習者の認知負荷について考える。認知負荷の指標については、Paas ら [18] によれば、尺度法・心理生理学による手法（心拍数や眼球運動などの観察）・二重課題法などがある。その一方で、Tarmizi・Sweller の例 [2] では、問題の解答時間から、問題の認知負荷を測定している。

Chang ら [43] は、プログラミング言語を教える講義において、スクリーン数が 1 画面での講義と 2 画面での講義での認知負荷と学習効果を比較している。彼らは、事後アンケートの結果と事前・事後テストの結果からこれらを考察している。事後アンケートによる認知負荷の測定は、尺度法にあたる。事後アンケートでは、「内容の難しさ」「説明のわかりにくさ」について尋ねた。これらはそれぞれ「内的認知負荷」、「外的認知負荷」を表す。

ところで、アンケートによる自己評価を用いたよる認知負荷の測定が、正確でないという批判がある。これに対しては、Gopher・Braune ら [7] が、作業負荷と自己評価による点数の間に強い関係があることを示している。また、Paas[17] や Pollock ら [12] のように、アンケートを用いた認知負荷の測定が行われている例がある。

本実験では、Chang らの手法と同様に、事後アンケートを行った。本実験におけるこれらのアンケート結果を両手法間で比較したところ、$p < 0.05$ で有意差がみられた項目がないこと、また、効果量が large のものもみられないことから、内的・外的認知負荷ともに両手法間での差が小さいものと推測される。

6.5.4 事後テストの成績・事後アンケートの結果と教授効率

事後アンケートの結果と、事後テストの成績から、提案手法による授業・従来手法による授業それぞれの教授効率を求めることを考える。Paas・van Merriënboer[19] [18] は、学習者の Mental Effort とパフォーマンスから、教授効率を求める図や公式

を提案している。また、Cerpa ら [35] は、教授効率を求めるのにこの公式を用いている。

この公式によれば、教授効率 E は以下の式で表される。

$$E = \frac{z_{\mathrm{Performance}} - z_{\mathrm{MentalEffort}}}{\sqrt{2}} \tag{6.1}$$

で表される。ここで、$z_{\mathrm{Performance}}$ はパフォーマンスの z スコア、$z_{\mathrm{MentalEffort}}$ は Mental Effort の z スコアを表す。また、$\sqrt{2}$ は、直線 $ax + by + c = 0$ と点 (x_1, y_1) との距離 d の公式

$$d = \frac{|ax_1 + by_1 + c|}{\sqrt{a^2 + b^2}} \tag{6.2}$$

に由来する。すなわち、E の絶対値は、座標上の点 $(z_{\mathrm{Performance}}, z_{\mathrm{MentalEffort}})$ と直線 $z_{\mathrm{Performance}} = z_{\mathrm{MentalEffort}}$ との距離を表す。この直線は、$E = 0$ を表す。さらに、$E > 0$ のとき、教授効率は高く、$E < 0$ のとき教授効率が低いといえる。

内的認知負荷と事後テストの成績

各セクションにおいて、授業の説明のわかりにくさすなわち外的認知負荷と、事後テストの成績の関係（空欄補充・コード読解・コード記述）を、図 6.6 に示す。また、各セクション・問題種別における、効率 E の値を表 6.8 に示す。

効率 E の値は、空欄補充問題とコード記述問題では、従来手法のほうが高効率となっている。一方で、コード読解問題では、提案手法のほうが高効率となっている。

このことから、内的認知負荷の観点において、コード読解問題では、提案手法の教授効率が高いと考えられる一方、空欄補充やコード記述問題においては効率が悪いと考えられる。

表 6.8 内的認知負荷に対する効率

内的認知負荷	実験手法			従来手法		
セクション	空欄補充	コード読解	コード記述	空欄補充	コード読解	コード記述
1	0.91	0.49	1.36	-1.35	-0.72	-2.02
2	-1.03	0.04	-1.74	0.70	-0.03	1.17
3	-1.06	-0.41	-1.25	1.57	0.62	1.85
4	-0.28	0.91	1.01	0.19	-0.61	-0.68
効率 E の平均	-0.37	0.25	-0.16	0.28	-0.19	0.08

外的認知負荷と事後テストの成績

各セクションにおいて、授業の説明のわかりにくさすなわち外的認知負荷と、事後テストの成績の関係（空欄補充・コード読解・コード記述）を、図 6.7 に示す。また、各セクション・問題種別における、効率 E の値を表 6.9 に示す。

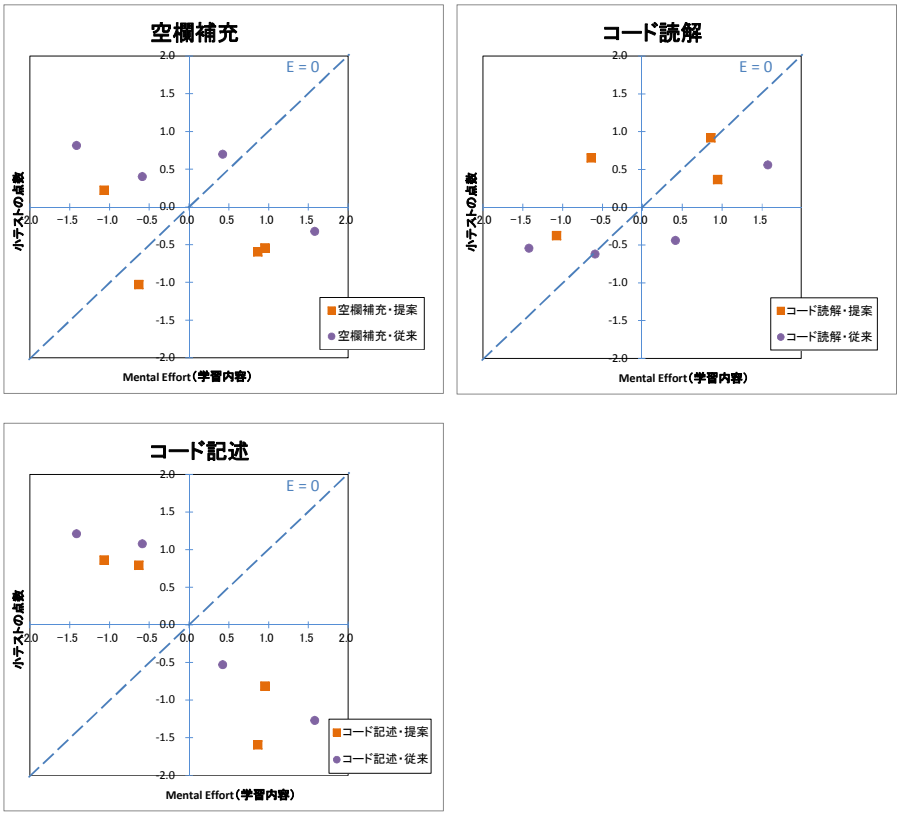

図 6.6　内的認知負荷と事後テストの成績

効率 E の値は、空欄補充問題とコード記述問題では、従来手法のほうが高効率となっている。一方で、コード読解問題では、提案手法のほうが高効率となっている。

このことから、外的認知負荷の観点において、コード読解問題では、提案手法の教授効率が高いと考えられる一方、空欄補充やコード記述問題においては効率が悪いと考えられる。

認知負荷と事後テストの成績のまとめ

コード読解問題は、プログラムの動作メカニズムを理解しているかを尋ねる問題である。これらの結果から、内的認知負荷・外的認知負荷両方の観点において、提案手法が「プログラムのメカニズムを理解させる」上で役立つ可能性が示唆された。

その一方で、単純な知識を問う空欄補充問題では、提案手法の効率が高いわけではないことが示された。これは、知識の伝達において提案手法が必ずしも有効でないことを示している。同様に、コード記述問題においても効率が高くない。これは、今回の模擬授業時間内では受講者がコード記述の練習を行っておらず、準備が不十分であった可能性が考えられる。

コード読解問題に対して、他の 2 つの問題の効率が提案手法で低くなってしまった原因としては、学習者の学習リソースが限られていることが考えられる。今回の実験では、限られた時間により学習効果について測定した。そのため、もし 1 つの項目が効果的になってしまった場合に他の項目についての学習が疎かになってしまう可能性がある。今回の実験では、プログラムの動作メカニズムについては理解し、記憶に残ったものの、例えば語句の記憶といった内容については学習の順位が下がり、結果として学習が阻害されてしまったのではないかと考えられる。

これが事実であるとすれば、学習者には十分な時間を与えて学習させること、状況に応じて他の適切な学習手段を与えること（例えば、語句の知識については、授業前の事前課題として与えるなど）することなどが解決策となりえる。

以上を踏まえ、提案手法、すなわち CodEx を利用することが、プログラミング学習者にプログラムの動作メカニズムを教授する上で有用であると考えられた。

表 6.9 外的認知負荷に対する効率

外的認知負荷	実験手法			従来手法		
セクション	空欄補充	コード読解	コード記述	空欄補充	コード読解	コード記述
1	0.29	-0.13	0.74	-0.43	0.20	-1.10
2	-0.31	0.77	-1.01	0.21	-0.52	0.68
3	-0.76	-0.11	-0.95	1.13	0.17	1.41
4	-0.41	0.78	0.88	0.28	-0.52	-0.59
効率 E の平均	-0.30	0.32	-0.09	0.30	-0.17	0.10

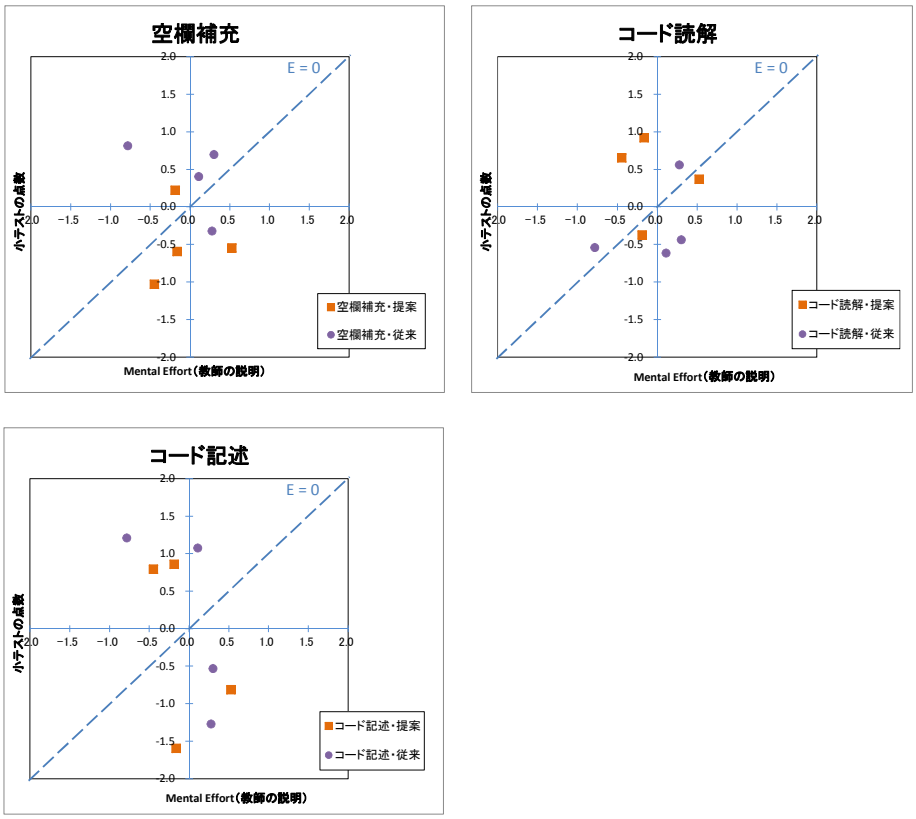

図 6.7　外的認知負荷と事後テストの成績

6.5.5　実験終了時アンケート

実験終了時アンケートにおける被験者の回答からは、CodEx がプログラミングの講義を聞く場合に有効である可能性が示された。しかしながら、学習内容の記憶に関して、よいと回答した被験者の数は、従来手法・提案手法で同数となった。これは、事後テストにて、提案手法・従来手法のどちらが学習内容を記憶する上で有効であったか、有意な結果を得られなかったことと一致する。

6.5.6　考察のまとめ

本実験は、被験者数が少数であることもあり、提案手法と従来手法のどちらがよいかを一般化して示すものではない。そのため、より詳細には追加実験が必要であると考えられるが、本研究は CodEx の効果を測定するための初期実験として有益なものと考えられる。

現時点での結果をまとめると、CodEx はプログラミングの講義のうち、プログラムの動作メカニズムを教える上で有効である可能性が示された。事後テストにおけるパフォーマンスだけでなく、内的認知負荷・外的認知負荷との関係からも、効率が高いことが伺えた。その一方で、単純記憶に関する項目では、学習効果に差がみられないものと思われる。また、本実験では、コードを記述する練習を行っていないため、コード記述を行う訓練が別途必要と考えられた。

6.6　本章のまとめ

本章では、CodEx を利用した模擬授業の評価について述べた。後者の実験からは、CodEx はプログラミングの講義のうち、メカニズムを教える上で有効である可能性が示された。その一方で、単純記憶に関する項目では、学習効果に差がみられなかった。

第7章

結論

本章では、各章をまとめ、本論文の結論について述べる。

7.1　まとめ

7.1.1　Web プログラミング言語教育のためのプレゼンテーションソフトウェア「CodEx」の開発

第 2 章では、Web プログラミング言語教育のためのプレゼンテーションソフトウェア「CodEx」の開発について述べた。CodEx は、スライド上に表示されたコードボックスにプログラムのサンプルコードを示しながら、実行ボックス上にその実行結果を表示し、提示することができるプレゼンテーションソフトウェアである。また、コードボックス上のサンプルコードは、プレゼンテーションをしながらであっても編集することが可能である。また、実行ボックスには、編集後のサンプルコードの実行結果を示すことができる。

CodEx の要求仕様は、以下に示すものであった。

- C1: 教師がスライドを表示し、学習者に講義ができる
- C2: 1 画面の利用
- C3: スライド上に説明文や視聴覚教材を表示
- C4: コードボックスにサンプルコードを表示
- C5: 実行ボックスの表示
- C6: コードボックスや実行ボックスは 1 スライド内に複数配置できる
- C7: 実行ボックスにはラベル名をつけることができる
- C8: コードボックスには、実行先の実行ボックスのラベル名を設定できる

- C9: コードボックスには、言語の種類を設定できる
- C10: 実行ボックスにサンプルコードの実行結果を表示できる
- C11: コードボックスを逐次編集できる
- C12: コードボックスの編集結果を、実行ボックスに反映できる
- C13: 実行ボックスには、複数のサンプルコードの実行結果を重ね合わせたものが表示できる
- C14: 学習者がスライド資料を手元で使って学習できる

機能と実装については、CodEx スライドの例を示しながら、CodEx スライドの構成要素である「コードボックス」と「実行ボックス」の機能と実装について述べた。

7.1.2 「CodEx」を用いた講義の評価

第 3 章では、前章にて開発した CodEx について、ユーザビリティ評価実験を行った。

7.1.3 スライド作成・編集ソフトウェア「CodEx GUI Editor」の開発

第 4 章では、CodEx GUI Editor（Editor）の開発について述べた。これは、CodEx プレゼンテーションの作成を容易に・グラフィカルに作成することを可能とするために開発された。Editor を用いることで、コードボックスや実行ボックスを含むスライドを WYSIWYG に作成することができる。

7.1.4 「CodEx GUI Editor」のユーザビリティ評価

第 5 章では、前章で開発した CodEx GUI Editor についての評価を行った。具体的には、Editor により，教員がスライドを容易に作成可能であるかどうか，ユーザビリティ評価を行った。その結果、評価を行った機能については利用可能であることが示された。一方で、コードボックス・実行ボックスをスライド内に配置する機能については、アンケート結果から改善の必要性が指摘された。

7.1.5 CodEx を用いた講義の評価

第 6 章では、学生側の視点から CodEx を評価するため、模擬授業による CodEx の評価実験を行った。実験からは、CodEx はプログラミングの講義のうち、メカニズムを教える上で有効である可能性が示された。その一方で、単純記憶に関する項目では、学習効果に差がみられないものと思われた。

7.2 結論

第 2 章から第 6 章までの結果をもとに、本論文の結論を述べる。

本論文では、講義形式によるプログラミング教育をより効率的に行うためのプレゼンテーションシステム「CodEx」を提案し、その有効性を実証することを目的とした。

プレゼンテーションシステムは、「スライドを表示し、プレゼンテーションするための機能」と、プレゼンテーションソフトウェアを用いて講義を行う教師の準備を支援するため、「スライド・プレゼンテーションを作成・編集するための機能」が必要である。本論文では、これら両方を開発し、その機能の評価を行った。具体的には、第 2 章で、スライドを表示し、プレゼンテーションするための機能の開発と実装について述べた。また、第 4 章で、スライド・プレゼンテーションを作成・編集するための機能の開発と実装について述べた。

開発されたそれぞれのソフトウェアに対して、要求仕様の評価を行い、目的が達成されているかを調べた。第 3 章では、スライド・プレゼンテーションを作成・編集するための機能について、教師側からのユーザビリティ評価を行った。

さらに、第 5 章では、CodEx GUI Editor のユーザビリティ評価を行った。

これらの評価をまとめると、以下のようになる。

表 7.1　要求仕様項目の総合評価

項目	CodEx 表示	Editor	システム全体
スライドの表示 （プレゼンテーションソフトウェアの前提）	○	−	○
1 画面の利用	−	−	−
スライド上に説明文や視聴覚教材表示	○	−	○
コードボックスにサンプルコードを表示	○	○	○
コードボックスの最大行数は 20 行	○	−	○
実行ボックスの表示	−	○	○
コードボックスや実行ボックスは 1 スライド内に複数配置できる	−	○	○
実行ボックスにはラベル名をつけることができる	−	○	○
コードボックスには、実行先の 実行ボックスのラベル名を設定できる	−	○	○
コードボックスには、言語の種類を設定できる	−	○	○
実行ボックスにサンプルコードの 実行結果を表示できる	○	−	○
コードボックスを逐次編集できる	○	−	○
コードボックスの編集結果を、 実行ボックスに反映できる	○	−	○
実行ボックスには、複数コードの 実行結果を重ね合わせたものが表示できる	○	−	○
GUI インタフェースによりスライドが編集できる	−	○	○
WYSIWYG なスライド編集画面を表示できる	−	○	○
プレゼンテーションのプレビューを表示できる	−	○	○
コードボックスをスライド上に配置できる	−	△	△
（複数ボックスを自由に配置できる）	−	△	△
実行ボックスをスライド上に配置できる	−	△	△
外部リソースをスライド内に表示できる	−	○	○
プレゼンテーションファイルを読込・保存できる	−	○	○

さらに、第 6 章では、構築されたシステムを講義において導入した場合の学習効果について検討した。その結果、教師の説明を聞きながら、プログラムの動作メカニズムを理解することができることが推察された。

これらから、CodEx システムの開発によって、以下の効果が得られたとの結論を導く。

1. スライド上に表示されたコードボックスにプログラムのサンプルコードを示しながら、実行ボックス上にその実行結果を表示し、提示することができる

2. 教師は、CodEx を用いて、プログラムの動作メカニズムを伝えるためのスラ

イド・プレゼンテーションを作成できる

3. 受講者は、教師の説明を聞きながら、プログラムの動作メカニズムを理解することができる。

7.3　今後の課題と展望

7.3.1　今後の課題

今後の課題として、以下のものが挙げられる。

CodEx による学習効果の詳細な測定

本論文の第 3 章では、CodEx を用いた模擬授業を行い、検証を行った。しかしながら、調査の対象が限られていること、被験者の数が限られていることから、より一般的な結論を導くのは難しい。より詳細に検証実験を行うことで、CodEx による学習効果を測定する必要性があると考えられる。

多言語対応について

現状での対応言語は HTML・CSS・JavaScript であるが、PHP や Perl など、その他の Web プログラミング言語（主にサーバ向けプログラミング言語）への対応が考えられる（本論文では触れていないが、サーバを別途用意することで PHP には暫定的に対応している）。さらに、ブラウザ上でより多くのプログラミング言語を実行する手法として、ideone.com にて公開されている API を用いる方法が考えられる。この API を用いることで、本ツールにて実行可能なプログラミング言語として、40 以上の言語に対応させることも可能であると考えられる。

7.3.2　今後の展望

今後の展望として、以下のような項目が挙げられる。

受講者の手元での利用

CodEx プレゼンテーションは、Web を介して容易に公開可能である。受講者が、講義での説明とあわせて手元の PC でサンプルコードを実行するスタイルの学習を行うことができる。

上西・室田 [57] では、CodEx を用いた実際の授業における受講者に対するアンケート調査について報告している。この報告では、学生が CodEx プレゼンテーションを手元で利用する機会が多かったかについて質問を行っている。その結果、5 件法アン

ケートでの平均点が 4 点を超えていることから、受講した学生においては、CodEx が手元で利用されているものと思われる。

現時点では、CodEx プレゼンテーションを手元で用いることによる学習効果については未調査であるが、今後、このような調査が必要であると考えられる。特に、コードの編集・実行のログを取得するなどして、学習者の活動を明らかにすることが、CodEx を用いた講義をよりよくしていく上で重要と考えられる。

新時代のインタフェースへの対応

昨今のタブレット端末の普及により、今まで PC で一般的に用いられていたインタフェースの多くに修正を迫られている。本論文では、タブレット端末の利用を想定していないが、今後の教育の流れから、タブレット端末の利用を想定したインタフェースに対応する必要があるものと考えられる。

謝辞

本研究を進めるにあたり、多くの人々からのご支援と助言をいただきました。

指導教員である室田真男教授には学部時代から 8 年もの間、大変熱心なご指導を賜りました。博士論文執筆までに長い時間がかかってしまいましたが、完成に至ったのは、室田先生の熱意ある指導のおかげであると思います。

副指導教員である中川正宣教授には、リサーチアシスタントとしての 3 年半、仕事を通じての成長の場を提供していただきました。また、RA の仕事により、勉学のための経済的な支援を受けることができました。

論文審査教員の中山実教授、西原明法教授、西方敦博准教授には、お忙しい中、論文審査をお引き受けいただきました。また、本論文に対する適切なアドバイスをいただき、ありがとうございました。

6 年間所属した室田研究室の皆様には、その時々により多大なご助言・ご支援を受けることができました。特に、中鉢直宏氏・五味渕大賀氏・齋藤皓太氏・上田健太郎氏・仲谷佳恵氏には、公私にわたり、大変お世話になりました。

中川研究室の皆様には、公私にわたって、専門を越えた新たな経験をさせていただきました。特に、寺井あすか助教・鈴木秀輔氏と、現・東京大学総合文化研究科広域科学専攻の和嶋雄一郎特任助教には、研究や実験の実施にあたり、多くのお手伝いをいただきました。

本研究は、筆者の尚美学園大学非常勤講師時代の授業経験に着想を得て開発されたものです。尚美学園大学の西嶋尚史 元教授には、授業において様々なご指導・ご助言をいただきました。

本論文の第 2 章・第 6 章の執筆にあたっては、室田研究室 OB の INOSTROZA Cueva Luis 博士と東京大学教育学研究科学校教育教育高度化センターの植阪友理助教には多大なご助言をいただきました。なお、Luis・Elena 夫妻は「CodEx」の名付け親でもあります。

第 4 章・第 5 章の執筆にあたっては、Web サイトの「ドットインストール」様、「HTML クイックリファレンス」様のご快諾により実験を行うことができました。

最後に、私の勤務先である獨協医科大学 基本医学情報教育部門・情報基盤センターの坂田信裕教授・センター長をはじめ、山下真幸講師、蓼沼隆課長補佐、梅村博子技術員、大橋和也技術員、冨士山千晶技術員のご協力のもとに本論文を書き上げることができました。また、基本医学担当の吉田謙一郎副学長、基本医学連絡会議議長の石井清教授には、本論文執筆の執筆にあたり、ご理解をいただきました。

この場をお借りして、厚くお礼申し上げます。

本論文に関する研究発表

原著論文、査読あり

- 上西秀和，室田真男．
  Web プログラミング言語教育用 HTML プレゼンテーション ”CodEx ” のためのスライド作成・編集ソフトの開発と評価，
  教育システム情報学会誌 31(2),pp.172-184, 2014（第 4・5 章）

- Hidekazu KAMINISHI, Masao MUROTA.
  Development and Evaluation of a New Presentation Software Program (CodEx) for Teaching Programming Code.
  Research and Practice in Technology Enhanced Learning. 6(2), pp.83-106, 2011（第 2・3・6 章）

国際会議発表、査読あり

- Hidekazu KAMINISHI, Masao MUROTA. Development and Evaluation of a Presentation Software for Web Programming Language Teaching, Proc. of the 18th ICCE, pp. 397-401, Dec. 2010, Putrajaya, Malaysia.（第 2・3 章）

国内学会発表 (査読なし)

- 上西秀和，室田真男．プログラミング言語教育用 Web プレゼンテーションの作成・編集ツールの開発, 電子情報通信学会 信学技報, ET2011-104, pp.25-30, Mar. 2012.（三豊, 香川）（第 4 章）

- 上西秀和, 室田真男. Web プログラミング言語講義のための教育システムの開発, 電子情報通信学会 研究会 教育工学（ET）, 電子情報通信学会技術研究報告 教育工学, 電子情報通信学会, Vol. 110, No. 312, pp. 41-46, Nov. 2010.（目黒, 東京）（第 2・6 章）

- 上西秀和, 室田 真男．プログラミング言語教育のためのプレゼンテーションツールの開発と評価, 2010 年 日本教育工学会 第 26 回全国大会, 2010 年 日本教育工学会 第 26 回全国大会講演論文集, 日本教育工学会, pp. 875-876, Sep. 2010.（名古屋, 愛知）（第 2・7 章）

- 上西秀和, 室田 真男. Web プログラミング言語教育のためのプレゼンテーションツールの開発, 電子情報通信学会 研究会 教育工学（ET）, 電子情報通信学

会技術研究報告, 電子情報通信学会, Vol. 110, No. 67, pp. 25-30, May. 2010.（草津, 滋賀）（第 2 章）

総説、査読なし

- 上西秀和.
  レビュー：プログラミング言語学習のための教育と教育システム，獨協医科大学 基本医学年報 2(1), pp.47-56, 2013（第 1 章の一部）

本論文以外での研究業績

国際会議発表 、査読あり

- Hidekazu KAMINISHI, Nobuhiro SAKATA, Masaki YAMASHITA. A Practice of PBL Training “Communications in Disaster” for Premedical Students with Twitter, Proceedings of the 1st IEEE Region 10 Humanitarian Technology Conference, pp.110-115, Aug. 2013, Sendai, Japan.
- Hidekazu KAMINISHI, Nobuhiro SAKATA, Masao MUROTA. Development of an HTML5 Presentation Software with Camera and Web Sharing Functions, Work-in-Progress Poster Proceedings of the 20th International Conference on Computers in Education ICCE 2012, pp.34-37, Nov. 2012, Singapore, Singapore.
- Hidekazu KAMINISHI, Masao MUROTA. Development and Evaluation of Presentation Software with Fisheye View Function for Programming Language Lecture Course, Proceedings of ED-MEDIA 2012, pp. 931-940, Jun. 2012, Denver, CO, USA.
- Hidekazu KAMINISHI, Shusuke SUZUKI, Asuka TERAI, Masanori NAKAGAWA. Construction of Japanese Search Engine Based on Computational Model of Inductive Reasoning, Proc. of the 19th ICCE, pp.125-127, Nov. 2011, Chiang Mai, Thailand.
- Hidekazu KAMINISHI, Masao MUROTA. Development of Multi-Screen Presentation Software, Proc. ED-MEDIA 2009, pp. 3934-3939, Jun. 2009, Honolulu, HI, USA.

国内学会発表 、査読なし

- 上西秀和，大釜徳政，山下真幸，鈴木純恵，坂田信裕．LMS を用いた授業評価アンケート環境構築と移行．第 8 回医療系 e ラーニング全国交流会講演要旨集，pp.20-21，15，Mar．2014（豊明，愛知）
- 山下真幸, 上西秀和, 坂田信裕. ICT（情報通信技術）を活用した新しい授業デザインの検討, 第 41 回 獨協医学会, In Dokkyo Journal of Medical Sciences, 41(2), pp.201, 7, Dec.2013（壬生, 栃木）

- 上西秀和, 坂田信裕, 山下真幸. 医療系大学における学内無線 LAN 環境と今後のモバイル端末を活用した授業形態について，教育システム情報学会（JSiSE）2012 年度特集研究会研究報告，vol.27, no.7, pp.92-97，16, Mar.2013（山口，山口）
- 坂田信裕，山下真幸，上西秀和．情報共有レベルの違いを体験することで学ぶ遠隔医療，日本遠隔医療学会スプリングカンファレンス 2013 抄録集，p13，15th，Feb，2013（文京，東京）
- 上西秀和, 坂田信裕, 山下真幸. 学内無線ネットワークの現状と今後の e-learning での活用に対応した環境について, 第 7 回医療系 e-Learning 全国交流会講演要旨集, pp.18-19, 12,Jan.2013（徳島, 徳島）
- 上西秀和, 坂田信裕, 山下真幸. ICT（情報通信技術）を利用した本学の教育環境に関する考察, 第 40 回 獨協医学会, In Dokkyo Journal of Medical Sciences, p.136, 1,Dec.2012（壬生, 栃木）
- 坂田信裕, 山下真幸, 上西秀和. 遠隔医療体験型授業から学ぶ医療における ICT 利活用力, 医療情報学, 3-C-2:一般口演 33, 32(Suppl.) pp.954-955, 17th,Nov.2012.（新潟, 新潟）
- 岡崎雄祐, 上西秀和, 室田真男. HTML5/CSS3 を利用した力学教育向けプレゼンテーション教材の開発, 2012 年 日本教育工学会 第 28 回全国大会 講演論文集, P2a-SCS-14, pp.651-652, Sep. 2012.（長崎, 長崎）
- 坂田信裕, 山下真幸, 上西秀和, 蓼沼隆, 冨士山千晶, 大橋和也, 梅村博子. 2 箇所のコンピューター教室を利用した双方向性連携授業環境の構築とその応用, 平成 24 年度教育改革 ICT 戦略大会資料, 公益社団法人私立大学情報教育協会, 大会発表 B-2, p.128-129, Sep. 2012.（千代田, 東京）
- 坂田信裕, 山下真幸, 上西秀和, 松野健二郎. カメラ付きタブレット端末の画面ミラーリング機能を利用した実習時の手元情報共有, 第 44 回日本医療教育学会大会予稿集, O9-II-4, p.88, Jul. 2012.（横浜, 神奈川）
- 上西秀和, 室田真男. 説明箇所を拡大表示可能なプログラミング教育向けプレゼンテーションツールの開発, 2011 年 日本教育工学会 第 27 回全国大会 講演論文集 pp.41-44, Sep. 2011. （八王子, 東京）
- 上西秀和, 室田真男. 複数画面を用いたプレゼンテーションツールの開発と評価, 日本教育工学会 第 25 回全国大会 講演論文集, pp. 525-526, Sep. 2009.（文京, 東京）
- 上西秀和, 室田真男. 複数画面を用いたプレゼンテーションツールの開発, 電子情報通信学会 2009 年総合大会 情報・システムソサイエティ総合大会特別号, p. 109, Mar. 2009.（松山, 東京）

- 上西秀和, 室田真男. チャットと情報共有ボードの併用による Web ブレーンストーミングツールの開発, 電子情報通信学会 2007 年総合大会 情報・システムソサイエティ総合大会特別号, Vol. ISS-P-54, p. 72, Mar. 2007.（名古屋, 愛知）

参考文献

[1] Ranganathan A. and Sicking J. (Eds.). File API W3C working draft 25 October 2012.
http://www.w3.org/TR/FileAPI/.

[2] Tarmizi R. A. and Sweller J. Guidance during mathematical problem solving. *Journal of Educational Psychology*, Vol. 80, No. 4, pp. 424–436, 1988.

[3] Roberts B. thenewboston.
http://thenewboston.org/index.php （2013/01/23 参照）.

[4] Carlisle M. C. Using youtube to enhance student class preparation in an introductory java course. Proceedings of the SIGCSE ' 10, pp. 470–474, 2010.

[5] Clark R. C. and Mayer R. E. *e-Learning and the science of instruction.* Pfeiffer, San Francisco, 2011.

[6] Buck D. and Stucki D. J. Jkarelrobot: a case study in supporting levels of cognitive development in the computer science curriculum. In *Proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education (SIGCSE '01)*, pp. 16–20, 2001.

[7] Gopher D. and Braune R. On the psychophysics of workload: why bother with subjective measures? *Human Factors*, Vol. 26, No. 5, pp. 519–532, 1984.

[8] Raggett D. HTML slidy: Slide shows in HTML and XHTML.
http://www.w3.org/2006/05/Slidy-XTech/slidy-xtech06-dsr.pdf.

[9] dotinstall.com. ドットインストール.
http://dotinstall.com （2013/02/14 参照）.

[10] Mayer R. E. *Multimedia Learning.* Cambridge University Press, New York, 2001.

[11] Mayer R. E. and Sims V. K. For whom is a picture worth a thousand words? extensions of a dual-coding theory of multimedia learning. *Journal of Educational Psychology*, Vol. 86, No. 3, pp. 389–401, 1994.

[12] Pollock E., Chandler P., and Sweller J. Assimilating complex information. *Learning and Instruction*, Vol. 12, No. 1, pp. 61–86, 2002.

[13] Briggs L. J. (Ed.). *Instructional design: Principles and applications*. Educational Technology Publications, Englewood Cliffs: NJ, 1977.

[14] Mayer R. E. (Ed.). *The Cambridge Handbook of Multimedia Learning*. Cambridge University Press, New York, 2005.

[15] Lunt B. M. et al. Information technology 2008 curriculum guidelines for undergraduate degree programs in information technology, 2008. http://www.acm.org/education/curricula/IT2008（2013/12/18 参照）.

[16] Kuo F., Chang T., Hsu J., and Yu P. The learning effects of simultaneous dual-screen instructional presentation in programming language instruction. In *Proceedings of the 17th International Conference on Computers in Education (ICCE '09)*, pp. 856–863, 2009.

[17] Paas F. Training strategies for attaining transfer of problem-solving skill in statistics: A cognitive-load approach. *Journal of Educational Psychology*, Vol. 84, No. 4, pp. 429–434, 1992.

[18] Paas F., Tuovinen J. E, Tabbers H., and van Gerven P. W. M. Cognitive load measurement as a means to advance cognitive load theory. *Educational Psychologist*, Vol. 38, No. 1, pp. 63–71, 2003.

[19] Paas F. and van Merriënboer J. J. G. The efficiency of instructional conditions: an approach to combine mental effort and performance measures. *Human Factors*, Vol. 35, No. 4, pp. 737–743, 1993.

[20] The Joint Task Force for Computing Curricula 2005. Computing curricula 2005 the overview report, 2005. http://www.acm.org/education/curric_vols/CC2005-March06Final.pdf（2013/12/18 参照）.

[21] Joint IS 2010 Curriculum Task Force. Is 2010 curriculum guidelines for undergraduate degree programs in information systems, 2010. http://www.acm.org/education/curricula/IS（2013/12/18 参照）.

[22] Kaminishi H. and Murota M. Development and evaluation of a new presentation software program (CodEx) for teaching programming code. *Research and Practice in Technology Enhanced Learning*, Vol. 6, No. 2, pp. 83–106, 2011.

[23] Kaminishi H. and Murota M. Development and evaluation of presentation software with fisheye view function for programming language lecture course. In

Proceedings of the World Conference on Educational Multimedia, Hypermedia and Telecommunications (Ed-Media'12), pp. 931–940, 2012.

[24] Kaminishi H., Sakata N., and Murota M. Development of an HTML5 presentation software with camera and web sharing functions. In *Work-in-Progress Poster (WIPP) Proceedings of the 20th International Conference on Computers in Education (ICCE '12)*, pp. 34–37, 2010.

[25] HTMQ. HTML クイックリファレンス.
http://www.htmq.com/ （2013/02/14 参照）.

[26] Prezi Inc. Prezi.
http://prezi.com/index/3/ （2014/01/09 参照）.

[27] Cohen J. *Statistical power analysis for the behavior sciences (2nd ed.)*. Lawrence Erlbaum, Hillsdale.

[28] Lanir J., Booth K. S., and Tang A. Multipresenter: a presentation system for (very) large display spaces. In *Proceedings of the 16th ACM International Conference on Multimedia (MM '08)*, pp. 519–528, 2008.

[29] Lanir J., Booth K. S., and Findlater L. Observing presenters' use of visual aids to inform the design of classroom presentation software. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (SIGCHI '08)*, pp. 695–704, 2008.

[30] Sweller J. and Chandler P. Why some material is difficult to learn. *Cognition and Instruction*, Vol. 12, No. 3, pp. 185–233, 1994.

[31] Mahin L. Powerpoint pedagogy. *Business Communication Quarterly*, Vol. 27, pp. 219–222, 2004.

[32] Masinter L. The data URL scheme (rfc2397).
http://tools.ietf.org/html/rfc2397.

[33] Nelson L., Ichimura S., and Pederson E. R. Palette: a paper interface for giving presentations. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (SIGCHI '99)*, pp. 354–361, 1999.

[34] Kolling M. and Rosenberg J. Guidelines for teaching object orientation with Java. In *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE '01)*, pp. 33–36, 2001.

[35] Cerpa N., Chandler P., and Sweller J. Some conditions under which integrated computer-based training software can facilitate learning. *Journal of Educational Computing Research*, Vol. 15, No. 4, pp. 345–367, 1996.

[36] Computer Science Curriculum 2008: An Interim Revision of CS 2001. Computing curricula 2005 the overview report, 2008.
http://www.acm.org/education/curricula/ComputerScience2008.pdf
（2013/12/18 参照）.

[37] The Joint Task Force on Computing Curricula. Computer engineering 2004 curriculum guidelines for undergraduate degree programs in computer engineering, 2004.
http://www.acm.org/education/education/curric_vols/CE-Final-Report.pdf
（2013/12/18 参照）.

[38] The Joint Task Force on Computing Curricula. Software engineering 2004 curriculum guidelines for undergraduate degree programs in software engineering, 2004.
http://sites.computer.org/ccse/SE2004Volume.pdf （2013/12/18 参照）.

[39] Ayres P. and Sweller J. The split-attention principle in multimedia learning. In Mayer R. E., editor, *The Cambridge Handbook of Multimedia Learning*, pp. 135–158. Cambridge University Press, New York, 2005.

[40] Chandler P. and Sweller J. Cognitive load while learning to use a computer program. *Applied Cognitive Psychology*, Vol. 10, No. 2, pp. 151–170, 1996.

[41] Anderson R., Anderson R., Simon S. A., and et al. Experiences with a tablet PC based lecture presentation system in computer science courses. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '04)*, pp. 56–60, 2004.

[42] Endo S. and Murota M. Development of dual-screen presentation support tool with zooming user interface. In *Proceedings of the World Conference on Educational Multimedia, Hypermedia and Telecommunications (Ed-Media '10)*, pp. 871–876, 2010.

[43] Chang T., Hsu J., and Yu P. A comparison of single- and dual-screen environment in programming language: cognitive loads and leaning effects. *Educational Technology and Society*, Vol. 14, No. 2, pp. 188–200, 2011.

[44] Tantau T., Wright J., and Miletic V. Latex beamer class.
https://bitbucket.org/rivanvx/beamer/wiki/Home （2014/01/09 参照）.

[45] Quint V. and Carcone L. Amaya home page.
http://www.w3.org/Amaya/ （2013/03/21 参照）.

[46] Chang Y., Chang T., Hsu T., and Yu P. The impact of split-attention effect on dual-screen learning environment for programming language instruction. In

Proceedings of the World Conference on Educational Multimedia, Hypermedia and Telecommunications (Ed-Media '09), pp. 2110–2115, 2009.

[47] インセプト. IT 用語辞典「プログラミング言語」とは, 2001.
http://e-words.jp/w/E38397E383ADE382B0E383A9E3839FE383B3E382B0E8A880E8AA9E.html （2012/12/14 参照）.

[48] R. M. ガニェ, W. W. ウェイジャー, K. C. ゴラス, J. M. ケラー. インストラクショナルデザインの原理. 北大路書房, 京都, 2007.
鈴木克明 岩崎信監訳.

[49] 安藤雅洋, 植野真臣. e ラーニングにおけるタブレット PC を用いた書き込みの効果分析. 日本教育工学会論文誌, Vol. 35, No. 2, pp. 109–123, 2011.

[50] 情報処理学会一般情報処理（GE）教育委員会. 一般情報処理教育の知識体系 (GEBOK), 2008.
http://www.ipsj.or.jp/12kyoiku/J07/20090407/J07_Report-200902/9/J07-GE_GEBOK-200803.pdf （2013/12/18 参照）.

[51] 河村一樹. 一般情報処理教育（J07-GE）. 情報処理, Vol. 49, pp. 768–774, 2008.

[52] 栗原一貴, 五十嵐健夫, 伊藤乾. 編集と発表を電子ペンで統一的に行うプレゼンテーションツールとその教育現場への応用. コンピュータソフトウェア, Vol. 23, pp. 14–25, 2006.

[53] 高山文雄. Web を利用した CG 重視のプログラミング教育支援システムの開発. 平成 23 年度教育改革 ICT 戦略大会資料, pp. 240–241, 2011.

[54] 高山文雄. Web 環境を利用したプログラミング教育支援システムの開発. 平成 24 年度教育改革 ICT 戦略大会資料, pp. 206–207, 2012.

[55] 篠原義和, 宮武明義. C プログラミング演習のための e ラーニングシステムの試作. 電子情報通信学会技術研究報告 ET 教育工学, 第 111(473) 巻, pp. 13–18, 2012.

[56] 松本真吾, 野中美希, 太田剛, 酒井三四郎. 教師が作成したテストケースを用いたプログラミングの正誤判定によるプログラミング学習支援システム. 教育システム情報学会誌, Vol. 26, pp. 29–35, 2009.

[57] 上西秀和, 室田真男. プログラミング言語教育のためのプレゼンテーションツールの開発と評価. 日本教育工学会第 26 回全国大会講演論文集, pp. 873–874, 2010.

[58] 上西秀和, 室田真男. Web プログラミング言語教育用 HTML プレゼンテーション ”CodEx ” のためのスライド作成・編集ソフトの開発と評価. 教育システム情報学会誌, Vol. 31, No. 2, pp. 172–184, 2014.

[59] 情報処理学会. 情報専門学科におけるカリキュラム標準 J07. 情報処理学会, 東京, 2009.
http://www.ipsj.or.jp/12kyoiku/J07/J0720090407.html （2013/12/18 参照）.

[60] 新開純子, 宮地功. プロセスの重視と評価活動を取り入れた C プログラミング教育の効果. 教育システム情報学会誌, Vol. 26, pp. 16–28, 2009.

[61] 森田彦. ICT を活用したプログラミング教育の実践. ICT 活用教育方法研究, Vol. 14, pp. 26–30, 2011.

[62] 西谷匠, 杉山雄一郎, 樋山聡, 桑原恒夫. 誤答に対する教師のリアルタイムでのアドバイスを支援する e-ラーニングシステム. 電子情報通信学会論文誌 D, Vol. J91-D, pp. 1538–1549, 2008.

[63] 筧捷彦. 非情報専門学科カリキュラム標準（副専攻）情報処理学会第 70 回全国大会シンポジウム発表資料, 2008.
http://www.ipsj.or.jp/12kyoiku/taikai70sympo/j07secondMajor.pdf
（2013/12/18 参照）.