プログラムコーディング過程を記録した動画教材の作成作業を支援するインタフェース

梶並知記[†1]　安田光[†1]
井上亮文[†1]　市村哲[†1]

本稿では，プログラミング講義において，教授者のプログラムコーディング過程を記録した動画教材の作成作業を支援するインタフェース「ProgRec」を提案する．プログラミング教育では，学習者が教授者のコーディング技術を参考にすることの有効性が知られており，また動画教材は，学習者の学習支援に有効である．しかしながら，教授者にとって動画教材の作成は負担が大きいという問題がある．本稿で提案するシステムは，教授者のコーディング過程をキー入力に応じて静止画で記録し，それを結合することで動画を生成するが，タイプミスなどによって生じる不要な静止画を自動的に削除し，冗長な動画の生成を防ぐ．また，コーディング中に随時短い解説（コメント）の挿入を可能にすることで，コーディング後に改めて動画編集作業を行う必要がないほか，コーディングから動画作成まで，他のシステムを一切使わずに行える．被験者に提案システムを用いた動画作成と従来システムを用いた動画作成を行ってもらう，動画作成作業の負担の少なさについて検証する実験行った結果，提案システムを用いることで，従来手法と比較して容易に動画教材の作成が可能なことを示す．

An interface for supporting making video-based study materials that record coding process

Tomoki KAJINAMI,[†1] Ko YASUDA,[†1]
Akifumi INOUE[†1] and Satoshi ICHIMURA[†1]

This paper proposes a ProgRec which makes a video-based study material which records a lecturer's coding process, for supporting programming education. In programing education, the video-based study material which records the lecturer's coding process is effective for a learner because he/she can refer the lecturer's coding process to take it. However, the lecturer is hard to make the video-based study material. Proposed system saves a image of screen automatically according to input from the keyboard by the lecturer, and makes the video-based study material by jointing the images. The system deletes automatically unnecessary images that are saved by making a typo by the lecturer. Moerover, the system permits a user to input several comments for explaining a program code during coding. That is, the lecturer not has to edit the video-based study material again, and he/she can make it using the system only. We perform subjective experiments for investigating effectiveness of proposed system. Experimantal results show that subjects can make the video easly using proposed system compared to with exsisting system.

1. はじめに

本稿では，プログラミング講義において，教授者のプログラムコーディング過程を記録した動画教材の作成作業を支援するインタフェース「ProgRec」を提案する．

近年，教育現場では，講義の様子を動画コンテンツとして保存したり，Web を通して配信したりといった試みが行われている[6),7)]．プログラミング講義でも，他人が行うコーディング過程を見ることがプログラムの学習に効果があり[2)]，また動画教材による学習支援の試みが行われている[7),9)]．しかしながら，動画コンテンツ作成において，動画コンテンツ作成側の負担を軽減することがあまり考慮されていなかったり，ビデオカメラや複数のソフトウェアを用いた大規模システムを用いるものが多い[12)]．

本稿では，プログラム講義において，教授者が講義中に自身が行ったコーディング過程を無駄なく記録し，さらにコーディング中に随時簡単なコメントを挿入した動画の作成を容易に行えるインタフェース「ProgRec」を提案する．ここで，無駄なく記録するとは，教授者の単純なタイプミスとそれに伴う修正作業の場面や，コーディング作業を一時止めて学習者に口頭で解説など行う非コーディング時間を含めず記録することである．作成した動画は，学習者の復習のために用いられる．提案システムを用いることで，教授者は，講義を行いつつ復習用の動画教材を容易に作成できるようになる．動画を再編集する作業は必要ない．動画教材作成作業の負担の少なさについて，実際にコーディング過程を記録するタスクを従来システムと提案システムそれぞれで被験者に行ってもらった結果，提案システムを使用した場合，従来システムを使用した場合より容易に動画教材の作成が可能なことを示す．

†1 東京工科大学コンピュータサイエンス学部

本稿の構成は以下のとおりである．2 節で，プログラミング教育支援に関する従来研究について述べ，本稿の位置づけを明確にする．3 節で，プログラムコーディング過程を記録する動画教材作成における課題と解決手段について述べ，4 節で，ProgRec を実装する．5 節で ProgRec の有効性を被験者実験を通して行い，提案システムを利用した場合，従来システムと比べて教授者が容易に動画教材の作成が可能なことを示す．

2. プログラミング教育支援に関する従来研究

近年，ペアプログラミングの技法[1] に倣い，他者のプログラムコーディング過程を参考にするプログラミング学習の支援が提案されている[2),4)]．また，学習者に内省を促進させる学習環境が提案され，失敗学に基づいたプログラミング教育支援が提案されている[3)]．学習者がより手軽に利用でき，プログラムの内容やデバッグの方法が容易に理解できるように，Web を利用したり，動画を利用した教材が提案されている[7)–9)]．これらは，学習者を主な対象とした研究であり，本稿の提案とは異なる方向性である．

教授者と学習者両方を支援する研究として，アシスタントロボットを教授者と学習者の仲介役とし，学習者の状況を教授者に伝達し，また学習者の問題解決を支援する研究がある[10)]．また，講義内容とそれに対する学習者の反応と，学習者の行う演習内容とその評価結果の履歴を取得し，学習者と教授者で学習者の理解促進のきっかけとなる事象を共有し，学習者の学習を支援する研究がある[11)]．これらは，教授者と学習者のインタラクションを対象にした研究であり，本稿の提案とは異なる．

プログラムは，同じ結果を返すものであっても，ソースコードのバリエーションが多岐にわたる場合がある．教授者を支援する研究として，学習者の作成したソースコードを，教授者が学習者に望んだコーディング過程に応じて評価する手法が提案されている[5)]．本稿で提案する「ProgRec」は，教授者を支援の対象とする．教授者が望ましいと考えるコーディングを学習者に提示する過程が，特別な編集を施さず動画となり，学習者が講義後に復習に用いることを想定している．本稿は，教授者側の教材作成作業を支援することを主眼としており，学習者の学習効率，理解力の増加といった面は考慮しない．また，本稿の提案は，複数の情報リソース，複数の機材を効果的に組み合わせた講義コンテンツ作成の支援とは異なる．本稿で提案する「ProgRec」は，ディスプレイキャプチャソフトやビデオカメラを使用せずにコーディングの過程を記録し，また Web 教材として使いやすい FLV（Flash Video）形式で動画を作成する．

3. プログラムコーディング過程の記録

3.1 コーディング過程の記録支援の意義

教授者は一般的にプログラミングの熟練者であり，学習者がもっていない，コーディングに関するノウハウをもっている．また，学習者にとって，熟練者のプログラミング技術を見ることが，自身の学習の助けになる．そのため，講義中に教授者が自身の行ったコーディング過程と簡単なコメントを記録した動画を容易に作成できれば，それが学習者の復習に役立つ教材の作成になると考える．

3.2 動画教材作成時の課題

動画教材を作成するにあたって，不便と感じるところ，通常のビデオ録画や従来システムによる動画教材の作成を行ったことがある教授者に対するヒアリングを行った．その結果，明らかになった課題は以下の 3 つに大別できる．

課題 1　コーディングしているかどうかに関わらず録画され続けたり，キーボードからの入力に応じて録画される場合でも，タイプミスを含んだ冗長な動画が作成されてしまう．

課題 2　コードの重要部分を簡単に説明するコメントの挿入といった，編集作業の負担が大きい．

課題 3　事前に録画用システムを準備したり，講義中に動画の作成に注力したりしなければならない．

これらは，教授者にとって動画作成作業の負担が大きく，現状より容易に動画作成作業ができることが望ましいことを意味している．これらの課題を解決することで，より容易に動画作成が可能になると考える．

3.3 インタフェースに要求される機能

プログラムコーディング過程を記録した動画教材の作成作業を支援するインタフェースには，3.2 節で述べた 3 つの課題を解決する機能が求められる．5 節で述べる従来システムに，ユーザのキーボード入力に応じて静止画を撮影してそれらの静止画を結合することで動画を作成するものがある．本稿で提案するインタフェースも，ユーザのキーボード入力に応じて静止画を作成，静止画を結合して動画を作成するが，3.2 節で述べた 3 つの課題それぞれに対応した機能を提案，実装する．

タイプミス自動削除機能　タイプミスの修正操作を行った静止画を削除する機能である．これにより，3.2 節の課題 1 を解決する．

コメント随時挿入機能　コーディング作業中，任意の箇所に簡単なコメントを挿入する機能

である．これにより，3.2 節の課題 2 を解決する．

コーディング・動画作成一体化機能　コーディングと動画作成を同一のインタフェース上で可能にする機能である．これにより，3.2 節の課題 3 を解決する．

タイプミス自動削除機能により，タイプミスが記録された余分な映像をカットする手間を省くことができる．これは，カーソルキーの位置を変更していない状態での BackSpace キーの入力に応じて，録画した静止画を削除することで実現する．

コメント随時挿入機能により，動画撮影後に別途システムを利用して動画編集をしたり，コーディングを最後まで終えてから改めてコメント挿入を行ったりする必要がなくなる．これは，インタフェース上に，コードの入力欄とは別にコメント入力欄を設け，コメントが必要な部分のコーディングが終わりしだい，すぐにコメントを挿入可能にすることで実現する．入力されたコメントは，コメントが必要な部分のすぐ上または下に表示されるよう，動画作成時に自動的に画像に挿入される．また，効率的なコーディング技術の 1 つとして，コードの一部を複製してから部分的に修正することがあるが，コメント随時挿入機能は，学習者へコピーと貼り付けを行ったことを伝えるコメントを，自動で挿入する．これは，コピー範囲選択操作，コピー操作，貼り付け操作に応じて，自動的に，コピーして貼り付けた旨のコメントを挿入することで実現する．

コーディング・動画作成一体化機能により，動画作成のための事前準備や，別のシステムを同時に利用して動画作成に注力するといった必要がなくなる．これは，プログラム作成によく使われる統合開発環境を模して，同一インタフェースでコーディング（コンパイル含む）と，ワンボタンで動画作成（別途編集作業が不要）できるようにインタフェースをデザインすることで，実現する．

4. ProgRec の実装

3.3 節で述べた機能を備えた，プログラムコーディング過程を記録した動画教材の作成作業を支援するインタフェース，ProgRec を計算機上に C#で実装した．**図 1** は，ProgRec を利用しプログラムをコーディングしている様子である．提案システムは，プログラミングに使用する統合開発環境を模したインタフェースデザインとなっており，ProgRec 上でソースコードの入力，コンパイルが行える．タイプミス自動削除機能は，ユーザが特にその利用を意識しなくても機能する．コメント随時挿入機能は，コードの任意の箇所を選択後，解説コメント入力部分に記入し，コメント挿入ボタンを押すことで機能する．コーディング・動画一体化機能は，動画作成ボタンを押すだけで機能し，自動的にコメント付きの動画が作成される．

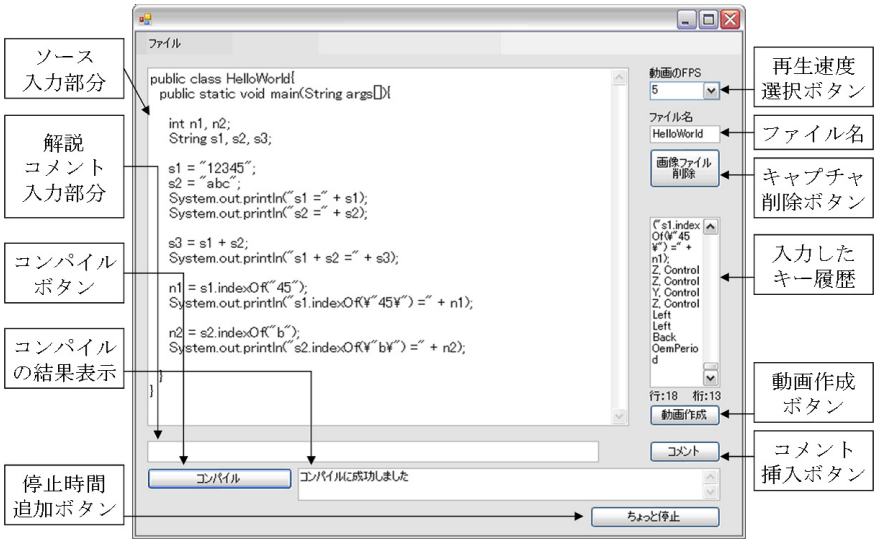

図 1　ProgRec 上でのプログラムコーディング
Fig. 1　Coding on the ProgRec

作成した動画は，FLV 形式で保存され，一般的な Web ブラウザ（Internet Explorer や FireFox）上で再生でき，Web 配信などを利用した遠隔教育現場での利用も容易である．

図 2 は，ユーザが何も明示的にコメント挿入操作を行わなかった場合でも，完成した動画にソースコードのコピー操作が行われた旨が表示される例である．コピーする範囲を選択する操作の映像が流れた後，自動的にコメントが表示される．**図 3** は，ユーザがソースコードの任意の箇所にコメントを挿入した結果，作成した動画で，挿入したコメントが表示される例である．ユーザがコメントを挿入した箇所のコーディングの表示に続き，自動的にコメントが表示される．

5. 評 価 実 験

5.1 目的と準備

実験の目的は，教授者が ProgRec を利用することで，従来手法と比較して容易に動画教材の作成が可能になったか検証することである．そのために，コーディングを含む，動画作

図 2 コピー操作に関するコメント
Fig. 2 Comment on copy of code

図 3 コードの任意の箇所に関するコメント
Fig. 3 Comment on arbitrary code

業にかかる時間を比較する定量的な評価と，動画作成作業に対する負担の少なさを表す手軽度や，完成した動画に対する満足度といったユーザビリティに関する定性的な評価を行う．

通常のビデオカメラによる撮影や，単体での動画編集が不可能な画面キャプチャシステム（単に画面を録画できるだけのもの）などと比較して，提案システムが容易に動画教材が作成できることは自明であるため，本稿では，提案システムと類似する従来システム「Wink」[*1]と比較する．**図 4** は Wink のインタフェースである．キーボードから入力が行われるごとに任意ウィンドウのキャプチャが行われ，キャプチャ終了後に不要な静止画の削除や，任

*1 http:www.debugmode.com/wink/

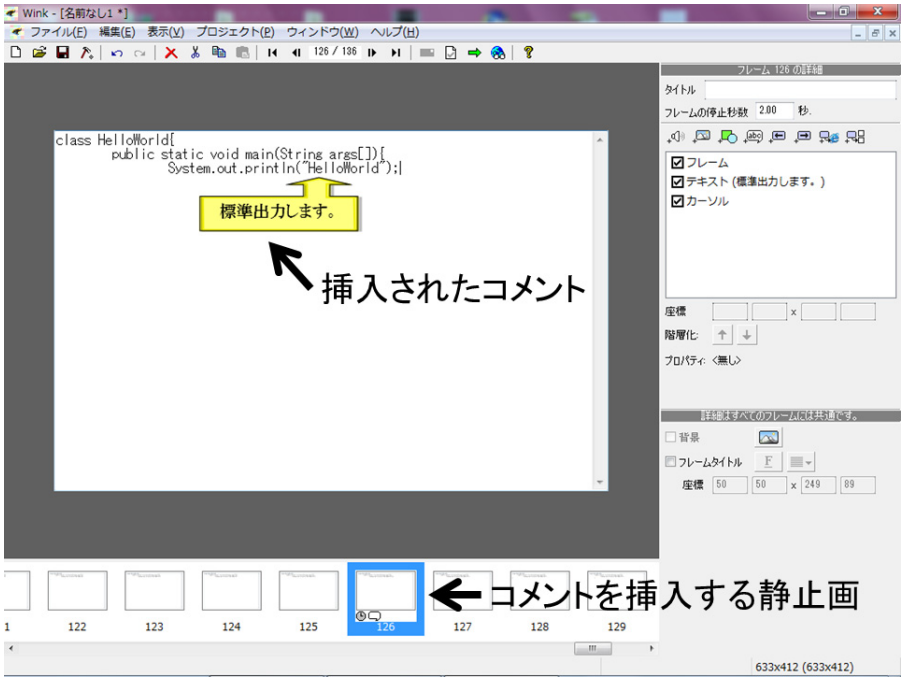

図 4 Wink による動画作成
Fig. 4 Video making by the Wink

意の静止画にコメント挿入ができる．最後に，静止画を結合して動画を作成する．

実験の手順は，以下の通りである．

(1) プログラムの基本要素である，逐次，反復，分岐を含んだ短いプログラムのコードを 2 種類（code-A，code-B）用意する．
(2) 被験者に，従来システムか ProgRec を利用してもらい，code-A か code-B のコードを入力してもらう．
(3) 被験者に，アンケートに回答してもらい，またヒアリングにも協力してもらう．
(4) 被験者に，(2) と異なるシステムを利用してもらい，(2) と異なるコードを入力してもらう．その後に (3) を行う．
(5) 被験者を変えて，(2) ～ (4) を行う．

code-A は 16 行からなり，タイプミスとその修正を行う箇所を 5 か所指定している．この指定の理由は，被験者がまったくタイプミスせずコーディングした場合に，タイプミス自動削除機能の評価ができなくなるからである．なお，指定箇所以外の自然なタイプミスは許可している．コメントを挿入する箇所は，コピー操作の箇所を除いて 5 か所指定している．コピー操作は，1 か所指定している．code-B は，22 行からなり，タイプミスとその修正を行う箇所の数，コピー操作の除きコメントを挿入する箇所の数は code-A と同じである．コピー操作は，2 か所指定している．

被験者は，プログラム講義で学習者への教育指導経験がある 6 名（A-F）である．使用するシステムの順序，コードの種類の順序は，被験者ごとに入れ替える．また，実験前に，HelloWorld プログラムを用いたシステムの操作練習を被験者に行ってもらう．

動画作成時間は 5 秒単位で記録し，アンケートは,「動画作成作業の手軽度」,「完成した動画に対する満足度」の 2 項目 5 段階（5：Good，1：Bad）評価のほか,「もし○○」（ならば）2 項目の評価がどう変化するかを予想する項目を設けている．○○の部分は，使用システムごとに異なっており，従来システムを使用した場合は「タイプミスをした場面が自動で削除されるなら」と「コーディングしながら随時コメントを入れるなら」であり，ProgRec を使用した場合は「タイプミスをした場面を手動で削除するなら」「コーディングが最後まで終わってからコメントを改めて入れるなら」である.「もし○○」の評価は，動画作成作業の手軽度に関しては,「より負担が少ない」「変わらない」「より負担が多い」，完成した動画に対する満足度に関しては,「より満足する」「変わらない」「より不満になる」の，それぞれ 3 つの選択肢から選んでもらう．また，アンケートでは，なぜそのような評価をしたのか理由についてできるだけ記入してもらい，記入された内容をもとにヒアリングを行って，より詳細な理由や記入されていない被験者の考えを取得する．

本実験の仮説をまとめると，以下のようになる．これら 3 つの仮説を満たせば，提案システムが従来システムと比較して容易に動画を作成できることになる．

仮説 1　提案システムを使用した場合の方が，動画作成時間が短い．

仮説 2　提案システムを使用した場合の方が，手軽度が高い．

仮説 3　提案システムを使用した場合でも，従来システムを使用した場合と同程度の満足度となる．

5.2　結果と考察

5.2.1　全体的な評価

表 1 は，被験者ごとに使用システムの順序，実行したタスク，アンケート結果をまとめたものである．被験者の括弧内には，使用システムと入力したコードの種類の組み合わせた順序を表している．PR は提案システムである ProgRec，Wk は従来システムである Wink を表し，ハイフン後の A または B は，それぞれ code-A，code-B を表している．例えば，被験者 A であれば，最初に ProgRec を使用し code-A を入力するタスクを実行し，次に Wink を使用し code-B を入力するタスクを実行したことを表している．動画作成時間は 5 秒単位で「分:秒」で表している．Wk が Wink，PR が ProgRec である．また，Wink 使用時は，コーディングと動画作成のための編集作業（タイプミスなど不要な箇所の削除や，コメントの挿入）が明確に分離しているため，作業全体にかかった時間だけでなくコーディングのみの時間も括弧内に表記している．手軽度は，動画作成作業の手軽度に関する 5 段階アンケートの結果，満足度は，完成した動画に対する満足度に関する 5 段階アンケートの結果を示している．それぞれ，Wink と ProgRec にわけて表記している．

表 2 は,「もし○○」2 項目についてのアンケート結果を手軽度に関するものと満足度に関するものにわけてまとめたものである．手軽度と満足度それぞれに関して，タイプミスをした場面を自動で削除されるか手動で削除するか（項目名：タイプミス自動削除），コーディングしながら随時コメントを入れるか入れられないか（項目名：コメント随時挿入），項目名の機能が「もしあるならば」または「もしないならば」とし，それぞれ「If あり」と「If なし」で表記しまとめている．手軽度に関して「より負担が少ない」ならば+1,「変わらない」ならば 0,「より負担が多い」ならば-1 で表している．満足度に関して「より満足する」ならば+1,「変わらない」ならば 0,「より不満になる」ならば-1 で表している．変化予想傾向は，これらの値を加算したもので，正の値であれば「より負担が少ない」または「より満足する」と予想する傾向で，値の大きさがその傾向の度合いである．負の値であれば「より負担が大きい」または「より不満になる」と予想する傾向で，値の大きさがその傾向の度合いである．

表 1 から，提案システムは従来システムと比較して，短時間で手軽に動画作成できていることがわかる．従来システムでは，コーディング終了後に，タイプミスの場面を探し出し削除する操作，コメント挿入箇所を探し出す操作，コメントの吹き出しを表示させる位置を決める操作が必要である．対して提案システムでは，これらの操作が不要であることが，表 1 に示す動画作成時間の差につながったと考える．これにより，仮説 1 を満たす．定性的な評価である手軽度に関しても，平均値が従来システムの 2.2 に対し，提案システムでは 4.3 と大きく向上しており，仮説 2 を満たす．また，完成した動画に対する満足度は，従来システムを用いて作成した場合とほぼ変わっていない．これは，提案システムを用いた場合

表 1 動画作成負荷に対するアンケート調査結果と動画作成時間
Table 1 Result of a questionnaire about effort for making a video and result of video making times

被験者	動画作成時間 (m:s)		手軽度		満足度	
	Wk	PR	Wk	PR	Wk	PR
A (PR-A, Wk-B)	15:20 (5:15)	4:30	4	3	5	4
B (PR-B, Wk-A)	10:10 (3:25)	8:00	2	5	4	4
C (PR-A, Wk-B)	30:55 (6:25)	7:35	2	4	2	4
D (Wk-B, PR-A)	40:45 (6:35)	6:20	3	5	5	5
E (Wk-A, PR-B)	15:40 (4:20)	9:25	1	4	4	4
F (Wk-A, PR-B)	21:45 (4:05)	8:25	2	5	5	5
平均	22:25 (5:50)	7:25	2.2	4.3	4.2	4.3

表 2 手軽度と満足度に関する変化予想
Table 2 Prediction about change of easiness and satisfaction

被験者	手軽度の変化予想				満足度の変化予想			
	タイプミス自動削除		コメント随時挿入		タイプミス自動削除		コメント随時挿入	
	If あり	If なし	If あり	If なし	If あり	If なし	If あり	If なし
A (PR-A, Wk-B)	+1	-1	-1	+1	+1	-1	0	-1
B (PR-B, Wk-A)	+1	-1	+1	-1	0	0	0	0
C (PR-A, Wk-B)	+1	-1	+1	-1	+1	-1	0	0
D (Wk-B, PR-A)	+1	-1	+1	-1	0	0	+1	+1
E (Wk-A, PR-B)	+1	-1	+1	-1	0	0	0	0
F (Wk-A, PR-B)	+1	-1	+1	+1	+1	-1	+1	0
変化予想傾向	+6	-6	+4	-2	+3	-3	+2	0

に，動画の見た目や内容の理解のし易さなど，従来システムを用いた場合と比較して著しく損なうことなく動画作成ができたことを示していると考える．これにより，仮説 3 を満たす．これらの結果から，提案システムは従来システムと比較して手軽に動画作成できているといえる．また，手軽度は，表 2 に示している手軽度の変化予想傾向と，一致している．満足度の変化予想傾向は，変化しないと予想した被験者と変化すると予想した被験者のグループにわかれた．唯一，被験者 A は，手軽度と満足度に関して，従来システムと比較して提案システムを低評価している．

5.2.2 節，5.2.3 節では，被験者のコメントから上記の結果に至った理由について述べる．

5.2.2 従来システムに対する肯定/否定的評価

従来システム Wink を用いた場合に被験者から得られた代表的なコメントを肯定/否定的評価で大別すると，以下のようになる．各コメント後の括弧内には，そのコメントと同じ趣旨のコメントをした被験者を示している．

- 肯定的評価
  - (1) コードの全体を見てから適切な箇所にコメント挿入できる.(A，B)
  - (2) コメント挿入位置の微調整やコメント内容修正などの自由度が高い.(B)
- 否定的評価
  - (3) 授業中に作業するには手間（時間・編集作業）が掛かる.(B，D，E，F)
  - (4) コーディング中にコメントを入れたいときにすぐにできない.(C)
  - (5) 手動で編集する際にミスが紛れる.(C)

(1)に関して，被験者 A は，復習用の動画を作成することを考えるならばコメントの質も重要で，Wink ではコード全体を見て後でコメントの追加や修正が行えるため，多少手間が掛かっても良い（講義中，練習問題などを学習者に解かせている間にできる程度＝十分手軽であると判断）とコメントしている．実際，手軽度の評価は 4 と高い．対して被験者 B は，自由度の高さを認めつつも，被験者 A と異なり，手間がかかるため手軽とはいえない評価をしている（評価値 2）．同じく手軽度を低評価した被験者 C は，プログラムコードの先頭から末尾まで順番にコーディングをしていく際に，学習者に示したい重要箇所に関してその場で説明とコメント挿入を行いたかったため，コーディングが最後まで終わらないとコメント挿入が不可能な点が不満だった．

総じて，従来システムは，作業時間の長さや編集操作の手間が大きなネックとなり，低評価となったと考える．完成した動画に対しては，被験者 C を除き,(特に問題なく）わかりやすい動画が作れたといった旨のコメントしている．被験者 C は,(5)に関連するが，編集作業で本来削除すべき箇所を削除し忘れるなど，自身が思っていた動画よりミスの多い動画が完成してしまったため不満だったとコメントしている．実際，満足度（表 1）は全被験者中最低の 2 と評価している．

手軽度や満足度の変化予想に関するコメントに目を向けると，以下のようになった．もし「タイプミスをした場面が自動で削除されるなら」手軽度が向上すると予想した被験者（全員）は,「削除する時間が短縮できるだろうから」,「削除するところを後で考えないで良いだろうから」といった旨のコメントをしている.「コーディングしながら随時コメントを入れるなら」手軽度が向上すると予想した被験者（被験者 A を除く）は,「コーディングしながら学習者に説明する箇所を説明しつつ流れ作業的にコメント挿入できるだろうから」,「学習者へ向けた説明で，気づいたことをその場でメモする感覚でコメント挿入できるだろうから（細かい説明を忘れなさそうだから)」,「コメント挿入箇所を，後で探す手間が省けるだろうから」といったコメントをしている．被験者 A は，コメントを挿入する箇所を意識し

つつコーディングを行うのが負担になるのではないかと予想している．

満足度の変化予想に関しては，変わらないと評価した被験者は，完成した動画の出来栄えは作業工程に依存しなからといった旨のコメントをしている．もし「タイプミスをした場面が自動で削除されるなら」満足度が向上すると予想した被験者は,「削除する予定だったところを残してしまうような手動編集のミスの心配がないだろうから」といった旨のコメントをしている.「コーディングしながら随時コメントを入れるなら」満足祖が向上すると予想した被験者は,「コメント挿入そのものや，コメントする細かい内容を忘れなさそうだから」といった旨のコメントをしている．

5.2.3 提案システムに対する肯定/否定的評価

提案システム ProgRec を用いた場合に被験者から得られた代表的なコメントを肯定/否定的評価で大別すると，以下のようになる．各コメント後の括弧内には，そのコメントと同じ趣旨のコメントをした被験者を示している．

- 肯定的評価
  - (1) 編集作業をせずに動画が作成できる.（A，B，D，E）
  - (2) 一定の速度でコーディング過程が表示される.（C）
  - (3) 見た目がシンプルな動画が作成できる.（D）
  - (4) 講義中に，無駄な手間が掛からない.（D）
  - (5) 自分が必要な時にすぐに使える.（F）
- 否定的評価
  - (6) すぐに気づいたタイプミスではなく暫く後で気づいたタイプミスの場面がそのまま残ってしまう.（A）
  - (7) コメント挿入位置の微調整ができない.（B，F）

総じて,（1）の理由により，高評価となっている．被験者 D は，練習タスク実行中から「便利だ」と呟いていた．タイプミス自動削除機能のおかげで編集不要となる点については，多少手間が掛かってもコメントの質を意識したい被験者 A にも肯定的に捉えられているが,（6）の理由により，総合的には手軽とは言えない評価をしている.（5）に関して，提案システムが統合開発環境を模しており，単体でコーディングから動画作成まで行えることが，被験者 F の手軽度の評価を上昇させることにつながっている．被験者 F は,（7）の否定的な評価をしつつも（5）の理由の方がはるかに重く，総合的には手軽度の評価値を 5 としている．

手軽度や満足度の変化予想に関するコメントに目を向けると，以下のようになった．もし「タイプミスをした場面を手動で削除するなら」手軽度が減少すると予想した被験者（全員）は,「復習動画教材にタイプミスの場面は不要であり，ミスした場面を削除する手間がかかるだろうから」,「講義中に，改めてタイプミスの場面を削除する編集を行うのは時間の無駄だから」といった旨のコメントをしている.「コーディングが最後まで終わってからコメントを改めて入れるなら」手軽度が向上するとした被験者 A と F について，その理由を問うと,「事前に用意した模範コメント一覧を見ながら正確にコメントを挿入し易くなるだろうから」（被験者 A）,「コード全体を見たときに，改めてコメントが必要だと思われる部分を確認してからコメント挿入できるだろうから」（被験者 F）といった旨のコメントを得た．動画教材の，特にコメントの質に対する意識が高い被験者の場合は，コーディング後に動画編集する流れを望んでおり，従来システムを使用した場合の肯定的評価（5.2.2 節，コメント（1））や，提案システムを使用した場合の否定的評価（6）（7）につながっていると考える．

満足度の変化予想に関しては，変わらないと評価した被験者は，5.2.2 節同様，完成した動画の出来栄えは作業工程に依存しなからといった旨のコメントをしている.「コーディングが最後まで終わってからコメントを改めて入れるなら」満足度が向上すると予想した被験者 D は,「改めてコメントの必要性を感じることがあるかもしれないから」と，被験者 F の手軽度の変化予想と類似する旨のコメントをしている．この被験者は，コメントの改めて挿入する場合でも，随時コメントを挿入する場合でも，どちらも満足度が向上すると予想し，満足度の評価は従来システムも提案システムも共に 5 としている．一見矛盾しているが，被験者 D は，それぞれのシステムを利用時に，システムの利点がそのままに，機能が追加される形でもし○○だったらと予想したため，どちらも満足度が高くなると予想することにつながった．満足度が減少すると予想した被験者 A は,「作成手順が分かりにくくなるだろうから」と，作成手順の伝達（提案システムを用いて作成した教材動画で学習者に伝えたい部分）の観点からコメントしている．しかし，被験者 A が提案システムを用いて動画作成を行った場合には，コメントの質に着目した評価をしている（5.2.2）．

5.2.4 提案システムの有効性

5.2.1〜5.2.3 節から，提案システムを用いれば，特に下記の要件を満たす場合に，従来システムと比較して手軽に動画教材の作成ができるといえる．括弧内は，主に有効となる機能を示している．

- 教授者が動画の質より作成時間を優先する場合（タイプミス自動削除機能，コメント随時挿入機能）
- 教授者が自身で行う動画編集作業のミスを回避したい場合（タイプミス自動削除機能）
- 教授者が単体システムでコーディングから動画作成まで行いたい場合（コーディング・

動画作成一体化機能）

提案システムは，細かく編集を重ねて見た目や内容に拘った動画を作成する場合より，講義中に短時間かつ単一システムでの動画作成を優先する場合に有効である．

6. おわりに

本稿では，プログラミング講義において，教授者のプログラムコーディング過程を記録した動画教材の作成作業を支援するインタフェース「ProgRec」を提案した．

従来の動画教材作成手法が抱える 3 つの問題点に対し，それらを解決する 3 つの機能,「タイプミス自動削除機能」「コメント随時挿入機能」「コーディング・動画作成一体化機能」を提案した．提案機能を備えた動画教材作成作業支援インタフェース ProgRec を実装し，プログラミング講義の教授者に利用してもらい，システムの評価を行った．

実験の結果，提案システムを利用する場合，類似した従来システムを利用した場合と比較して，容易に動画教材が作成できることが示された．しかしながら，簡単な操作でコメント挿入を行えるようにしたことと，無編集で動画教材を作成できるようにした代わりに，特にコメントの位置についての微修正が不可能な点が被験者の不満となる場合があることがわかった．

今後の課題として，挿入したコメントの位置の微調整を任意に可能とすることが挙げられる．また，本稿では，講義において学習者よりも教授者側の支援に重点をおいた提案を行ったが，今後は学習者側の支援に重点をおき，学習者からのフィードバックを考慮し，ある学習者が他の学習者の学習状況なども理解できるよう，システムの拡張を行っていく．

参 考 文 献

1) Beck, K. and Andres, C.: Extreme Programming Explained: Embrace Change, Addison-Wesley, 2nd Edition (2005).
2) Braught, G., Edy, L.M. and Wahls, T.: The Effects of Pair-Programming on Individual Programming Skill, *Proc. of the 39th SIGCSE technical symposium on Computer science education*(SIGCSE'08), pp.200–204 (2008).
3) 知見邦彦, 櫨山淳雄, 宮寺庸造: 失敗知識を利用したプログラミング学習環境の構築, 電子情報通信学会論文誌, Vol.J88-D-I, No.1, pp.66–75 (2005).
4) Fronza, I., Sillitti, A. and Succi, G.: An Interpretation of the Results of the Analysis of Pair Programming during Novices Integration in a Team, *Proc. of the 3rd International Symposium on Empirical Software Engineering and Measurement*(ESEM'09), pp.225–235 (2009).
5) 小西達裕, 鈴木浩之, 伊東幸宏: プログラミング教育における教師支援のためのプログラム評価機構, 電子情報通信学会論文誌, Vol.J83-D-I, No.6, pp.682–692 (2000).
6) 小薗和剛, 寺本明美, 秋山秀典: オンライン授業コンテンツ作成のためのオーサリングソフトウェア, 電気学会論文誌 A, Vol.125, No.8, pp.675–682 (2005).
7) 松山智恵子, 中島豊四郎, 石井直宏: プログラミング教育のための Web ベース教材の開発と評価, 電気学会論文誌 C, Vol.125, No.12, pp.1900–1905 (2005).
8) Pais, R. and Barros, J.P,: Use of Flash Movies for Teaching GUI Programming, *Proc. of the 10th annual SIGCSE conference on Innovation and technology in computer science education*(ITiCSE'05), p.390 (2005).
9) Simon, B., Fitzgerald, S., McCauley, R., Haller, S., Hamer, J., Hanks, B., Helmick, M.T., Moström, J.E., Sheard, J. and Thomas, L.: Debugging Assistance for Novices: A Video Repository, *SIGCSE Bull.*, Vol.39, No.4, pp.137–151 (2007).
10) 玉田春昭, 萩野晃大, 上田博唯: アシスタントロボットを用いたプログラミング教育支援システムの構築, 電子情報通信学会研究報告, Vol.110, No.108, pp.143–148 (2010).
11) 田口浩, 糸賀裕弥, 山本哲男, 高田秀志, 島川博光: プログラミング演習評価と講義反応を連携させた理解の契機の抽出, 電子情報通信学会論文誌, Vol.J91-D, No.2, pp.345–357 (2008).
12) 八重樫理人, 谷川晃, 守屋英樹, 玉田裕司, 神澤雄智, 三好匠, 相場亮: 講義コンテンツ自動生成システムの開発, 電子情報通信学会論文誌, Vol.J91-D, No.12, pp.2819–2832 (2008).