KeT-LMS のスマートフォンでの利用促進に向けた模索

長野工業高等専門学校　濱口 直樹[1],
群馬工業高等専門学校　碓氷 久[2],
山口大学教育学部　北本 卓也[3]

1　はじめに

インターネットや IT 技術の発展した現在において，E-Learning の重要性は多方面にわたって示されている．まず，E-Learning は地理的，経済的障壁を低減し，より多くの人々が高品質な教育資源へアクセスできるようにする．これは特に遠隔地や資源が限られている地域の学習者にとって重要である．次に，E-Learning は柔軟性を提供する．学習者は自分のペースで学習を進めることができ，時間や場所を選ばずに学習資源を利用できるため，ライフスタイルや仕事との両立がしやすくなる．また，様々な学習スタイルやニーズに対応したカスタマイズが可能で，個々の学習者の効果的な学習をサポートする．さらに，E-Learning は教育の持続可能性にも寄与する．物理的な教材の必要性が減り，教育過程のカーボンフットプリントが低減されるため，環境に優しい教育の形態と言える．また，継続的な技術革新により，教育ツールとしての機能が拡張され，より効果的な学習が実現可能である．最後に，パンデミックによって，対面の教育が制限された状況で，E-Learning システムは教育を継続するための重要な手段となった．この経験は，世界中の教育システムにおいて E-Learning の導入と拡張を加速し，教育アクセスの平等化を推進する大きな契機となっており，パンデミックが終了した今なお，E-Learning は広がり続けている．

本論文では，学習管理システムである KeT-LMS のスマートフォンにおける利用方法に焦点を当てる．KeT-LMS は，KeTCindy([3]) の開発者である高遠氏によって整備が進められている LMS(学習管理システム) であるが，キーボード形式の入力機能を持つ学習管理システムであり，ユーザーはキーボードのボタンをクリックすることで数字やアルファベットの入力を行う．しかしながら，スマートフォンなど画面の狭いデバイスを使用する際，ボタンのサイズが小さくなり押し間違いが生じる問題が存在している．これに対処するため，本論文では，より効率的でエラーの少ない入力方法として，フリック入力の導入を検討する．

また，本研究では KeT-LMS にフリック入力機能を組み込んだ新たなシステムの開発を行う．より正確には，フリック入力と従来のキーボード入力をユーザーが容易に切り替えられるようなシステムを開発する．また，フリック入力とキーボード入力の使用状況を記録するログ機能を実装し，これらのデータを将来的な入力方法の改善やユーザーの好みの分析に活用する．

新たに開発されるシステムには，最適なインターフェースを提供するための様々な機能も含まれる．例えば，問題に応じて必要な図を画面上に表示させたり，入力された解答や

[1]E-mail: hama@nagano-nct.ac.jp
[2]E-mail: usui@gunma-ct.ac.jp
[3]E-mail: kitamoto@yamaguchi-u.ac.jp

ログのデータをボタンのクリックだけでサーバーに送信する事が可能になる．

この研究により，KeT-LMS は画面の狭いデバイス上での操作性が改善し，ユーザーの利便性の向上が期待される．

2 KeT-LMS について

2020 年からのパンデミックの中，大学や高専においても教育を継続していくために，オンライン形式での講義の実施等のリモート対応が行われた．数学教育においては，講義がリモートでの対応となった場合でも，課題の配付，答案の提出，採点および返却という一連の流れが重要であり，これらをどのように行っていくかについて，様々な試みがなされた．

そのような中，少ないデータで迅速かつ容易に数式の送受を行うことを目的として開発されたのが KeTMath である．KeTMath では，KaTeX を利用した出力方法に基づく簡易数式表現を設定している．高遠氏によって設定された KeTMath の数式ルールの一例を以下に記す ([1])．

(1) 分数　$\frac{a}{b} \Longrightarrow$ `fr(a,b)`

(2) 掛け算　$ab \Longrightarrow$ `ab`　（`a*b` も可）

(3) べき乗　$a^b \Longrightarrow$ `a^(b)`　（b が 1 文字の場合は `a^b` も可）

(4) べき乗根　$\sqrt{a},\ \sqrt[3]{a} \Longrightarrow$ `sq(a), sq(3,a)`

(5) 三角関数　$\sin x,\ \sin^2 x \Longrightarrow$ `sin(x), sin(x)^2`

(6) 度　$60^\circ \Longrightarrow$ `60(deg)`

(7) 円周率　$\pi \Longrightarrow$ `pi`

(8) 対数関数　$\log x,\ \log_a x,\ \ln x \Longrightarrow$ `log(x), log(a,x), ln(x)`

(9) 積分　$\int x^2\,dx,\ \int_a^b x^2\,dx \Longrightarrow$ `int(x^2,dx), int(a,b,x^2,dx)`

(10) ブラケット　$\left[f(x)\right]_a^b \Longrightarrow$ `br(f(x),a,b)`

(11) 極限　$\lim_{x\to a} f(x) \Longrightarrow$ `lim(x,a,f(x))`

(12) 和　$\sum_{k=1}^{n} k^2 \Longrightarrow$ `sum(k=1,n,k^2)`

(13) 微分・偏微分　$\frac{dy}{dx},\ \frac{\partial z}{\partial x} \Longrightarrow$ `diff(y,x)`, `par(z,x)`

(14) 行列・行列式　$\begin{pmatrix} a & b \\ c & d \end{pmatrix},\ \begin{vmatrix} a & b \\ c & d \end{vmatrix} \Longrightarrow$ `mat(a,b;c,d)`, `det(a,b;c,d)`

(15) 下添字　$a_n \Longrightarrow$ `a_n`

KeTMath のこの数式ルールを学生が理解することにより，数式を含む演習問題の解答を，データ量の小さいテキスト形式で送受することができる．しかしながら，実際にはスマートフォンで利用することも多いため，当初は「√」や「π」といった特殊文字や全角

文字を用いて記述する学生も散見され，数式が正確に出力されないこともしばしばであった．このような状況に対処するため，KeTMath 独自のキーボードが作成され，数式の入力に用いることとした．このキーボードのみを利用した場合には，上記のような学生の誤入力も減少した．

その後，高遠氏によって，演習問題の入力から配付する問題ファイルの作成，さらには数式処理システム Maxima による自動採点も含めた答案の採点処理までをスムーズに行うことのできる KeT-LMS が開発され，実際の授業でも利用されている ([2])．これを用いると，授業中に出題した HTML 形式の演習問題（図１左）に対して，学生がテキスト形式で解答を提出すると，授業時間内でも解答状況を確認することが可能となっている．

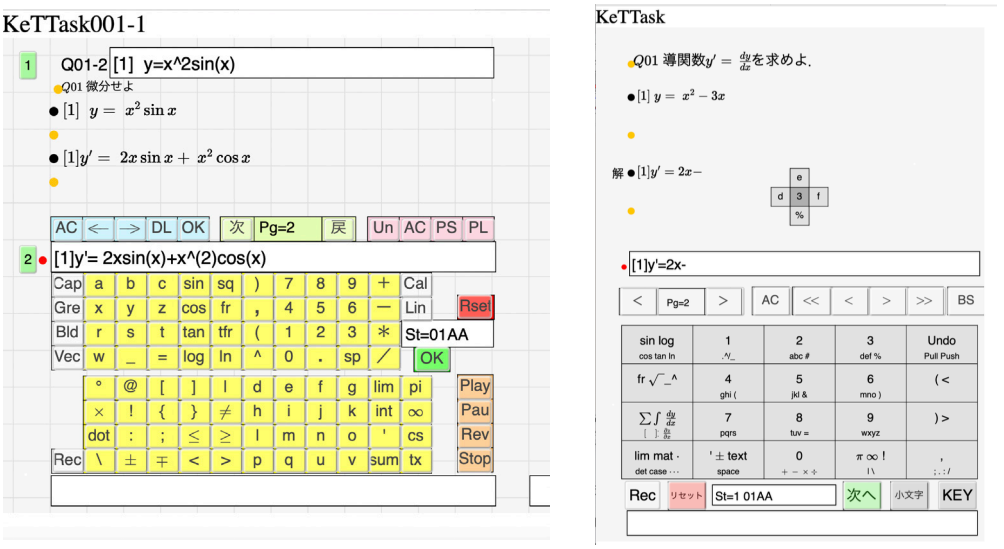


図１． 問題演習用 HTML ファイル：キーボード形式（左）とフリック入力形式（右）

一方で，学生の利用時の声や，問題演習用のシステムとしての教員側の要望としては以下のような点が挙げられる．

- キーボードのボタンを大きくすること
- 押し間違えた時の対処をスムーズにすること
- ピンチアウトによる拡大等が思い通りにできること
- 問題に関する情報を載せるスペースを確保すること
- 入力した解答に合わせてグラフが描かれる等のインタラクティブな要素を入れること

これらの課題への１つの対策として，フリック入力方式を採用した HTML ファイル（図１右）を作成した．これにより，ボタンの大きさや問題情報の掲載スペースを広げることができる．次節では，今回装備したフリック入力の詳細について述べる．

3 KeT-LMS でのフリック入力

ここでは，フリック入力機能を実装するために行ったことについて報告する．

KeT-LMS では，kettaskorgv.html をひな型として，各自が作成した問題を組み込んだ問題ファイルを作成する．入力方法以外はほとんど同じであるので，この kettaskorgv.html ファイルを直接書き換えることで，ひな型となるファイルを作成した．それを kettaskorgv.html の代わりに使うことで，フリック入力に対応した問題ファイルを作成できる．

フリック入力をする場合，ボタンを押したまま動かすことで入力する文字等を選択し，離すタイミングで入力するが，KeTCindyJS のボタンでそのようなことを実現するのは難しいと考え，当初はボタンを KeTCindyJS 領域外に HTML5 で作成していた．しかし，KeTCindyJS では数式記号等がきれいに表示できるが，HTML5 のボタン上に表記することは簡単にはできそうもなく，見た目があまりきれいではないものになっていた．そこで，ボタンではなく，クリックまたはタッチをする位置を認識して処理する形にした．

基本的には，CindyScript のコマンド mouse() によって位置を把握するが，mouse() は，ボタン上や入力ワク内では機能せず，その直前の値を保つ．マウスを使う場合はボタン上や入力ワク内以外で値を取得し続けているため利用できるが，スマートフォンなどのタッチパネル式の場合，次にタッチした場所がボタン上や入力ワク内だった場合には，その直前にタッチした位置情報を保持しているため，文字入力後にボタンや入力ワク内をタッチすると，再度文字入力がされてしまうという現象が起きる．そのため，HTML5 での位置も把握して利用することにした．

KeTCindyJS と HTML5 との間での情報の受け渡しとしては，html ファイル内で JavaScript により

cdy.evokeCS()

と書くことで，() 内のコマンドを CindyScript として実行することができる．

クリックまたはタッチのタイミングを取得するために，クリックの場合は mousedown，タッチの場合は touchstart を利用するが，そのタイミングでクリックまたはタッチした位置を把握するため，JavaScript で

ソースコード 1: 押したとき

```
1  document.addEventListener('mousedown', function(){
2    cdy.evokeCS("click1xj="+event.clientX);
3    cdy.evokeCS("click1yj="+event.clientY);
4    cdy.evokeCS("clickf0=1");
5    cdy.evokeCS("xoffset="+window.pageXOffset);
6    cdy.evokeCS("yoffset="+window.pageYOffset);
7    cdy.evokeCS("ctcheck1=1");
8  });
9
10 document.addEventListener('touchstart', function(){
11   cdy.evokeCS("click1xj="+event.touches[0].pageX);
12   cdy.evokeCS("click1yj="+event.touches[0].pageY);
13   cdy.evokeCS("clickf0=1");
14   cdy.evokeCS("ctcheck4=4");
15 });
```

と記述した．これにより，クリックまたはタッチをした場合に

clickf0=1

となり，HTML5 での位置を

(click1xj, click1yj)

のように，CindyScript 内で使える形で把握できる．なお，画面スクロールなどした場合に使う

(window.pageXOffset, window.pageXOffset)

の値も同時に取得する．

移動と離すタイミングについては，マウスの場合

ソースコード 2: 移動・離したとき（マウスクリック）

```
document.addEventListener('mousemove', function(){
  cdy.evokeCS("clickf0=2");
  cdy.evokeCS("ctcheck2=2");
});

document.addEventListener('mouseup', function(){
  cdy.evokeCS("clickf0=3");
  cdy.evokeCS("ctcheck3=3");
});
```

タッチパネルの場合

ソースコード 3: 移動・離したとき（タッチ）

```
document.addEventListener('touchmove', function(){
  cdy.evokeCS("clickf0=2");|\\
  cdy.evokeCS("ctcheck5=5");|\\
 });

document.addEventListener('touchend', function(){
  cdy.evokeCS("clickf0=3");
  cdy.evokeCS("ctcheck6=6");
});
```

とすることで，移動の場合

clickf0=2

離した場合

clickf0=3

となって，clickf0 の値によって状態を把握することができるようになる．

ctcheck1, ctcheck2, $\cdots$, ctcheck6

は，主に動作確認用に使用した．

追加として JavaScript で記述したものはこれだけであり，それ以外は全て，CindyScript での記述となる．

KeTCindyJS での位置

mouse()=(mouse()_1, mouse()_2)

と，HTML5 での位置

(click1xj, click1yj)

との関係は

click1xj $= a *$ mouse()_1 $+ b$

のようになっている．2ヶ所でのそれぞれの値から，a と b の値を求めることができる．

ソースコード 4: 2ヶ所の値から a と b を求める

```
1   if(acheckf==0,
2    if(or(ctcheck1==1,ctcheck4==4),
3     mxy1=mouse();
4     jxy1=[click1xj,click1yj];
5     acheckf=1;
6     ctcheck1=0;ctcheck4=0;
7    );
8   );
9
10  if(acheckf==1,
11   if(or(ctcheck1==1,ctcheck4==4),
12    ax=(jxy1_1-click1xj)/(mxy1_1-mouse()_1);
13    ay=(jxy1_2-click1yj)/(mxy1_2-mouse()_2);
14    acheckf=2;
15   );
16  );
17
18  drawtext([60,10],"acheckf="+acheckf);
19  drawtext([60,5],"[ax,ay]="+[ax,ay]);
20  drawtext([60,18],"mouse()="+mouse());
```

a の値はシステムや状態によっても不変であるようだったため，上のコードを実行して求めた値

x 座標については $a = 7.9268$　　y 座標については $a = -7.9268$

を使用した．システムや状態によって変化する b の値については，最初のクリックまたはタッチしたときの値から計算して求めることにした．

ソースコード 5: 座標の関係

```
1     changeax=7.9268;
2     changebx=click1xj-changeax*mouse()_1;
3     changeay=-7.9268;
4     changeby=click1yj-changeay*mouse()_2;
```

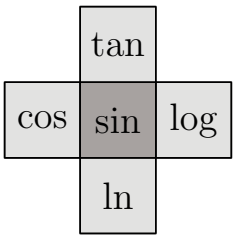
次に，フリック入力によるキー選択の仕組みについて述べる．

キー領域内をクリックした場合，clickf=1 となるようにし，そのときの位置 (click1x, click1y) を使って

ソースコード 6: 押したキーの判定

```
1  if(clickf==1,
2     select=1;
3     nx=ceil((click1x-kx0)/kdx);
4     ny=ceil(3+(ky0-click1y)/kdy);
5     butnum=ny*5+nx;
6  );
```

のようにして，どのキーかを判定し



のような十字形を表示する．押しながら動かすと clickf=2 となるようにし，そのときの位置 (click2x, click2y) を使って

ソースコード 7: フリックによる選択

```
1  if(clickf==2,
2     select=1;
3     if(click2x > click1x+2,select=4);
4     if(click2x < click1x-2,select=2);
5     if(click2y > click1y+2,select=3);
6     if(click2y < click1y-2,select=5);
7  );
```

のようにして移動方向を判定して色を変え，離したタイミングで clickf=3 とし

ソースコード 8: 入力

```
1  if(clickf==3,
2     funflg=1;
3      name=hairetu_butnum_3_select;
4      clickf=0;
5      butnum=0;
6  );
```

によって，入力文字を選択する．hairetu は hairetu1 から hairetu9 の 9 種類ある.「KEY」ボタンを押すことにより，中 3 列のキー表示を

数字を大きく表示するもの，　文字を大きく表示するもの，　数字のみ表示するものの 3 種類から選び,「KEY」の横のボタンを押すことで，文字を

英アルファベット小文字，　英アルファベット大文字，　ギリシャ文字

の 3 種類から選ぶ．例えば，hairetu1 は，数字を大きく表示する英アルファベット小文字のものであり，以下のようになっている．それぞれのキーについて

[[大きく表示するもの, 下に小さく表示するもの],
[十字形に表示する 5 つ],[入力される 5 つ]]
の順に並んでいる.

ソースコード 9: hairetu1

```
hairetu1=[
  [["sin log","cos tan ln"],["sin","cos","tan","log","ln"],["sin(?)","
     cos(?)","tan(?)","log(?)","ln(?)"]],
  [["1",".^/_"],["1",".","^","/","_"],["1",".","^","/","_"]],
  [["2","abc #"],["2","a","b","c","#"],["2","a","b","c","\#"]],
  [["3","def %"],["3","d","e","f","%"],["3","d","e","f","\%"]],
  [["Undo","Pull Push"],["Undo","","Pull","","Push"],["Delete()","
     Delete()","Pull()","Delete()","Push()"]],
  [["fr $\sqrt{\phantom{a}}$_^",""],["fr","tfr","$\sqrt{\phantom{a}}$
     ","^","_"],["fr(?,)","tfr(?,)","sq(?)","^(?)","_(?)"]],
  [["4", "ghi ("],["4","g","h","i","("],["4","g","h","i","("]],
  [["5","jkl &"],["5","j","k","l","&"],["5","j","k","l","\&"]],
  [["6","mno )"],["6","m","n","o",")"],["6","m","n","o",")"]],
  [["( <",""],["(","{","[","<","$\leq$"],["(","\{","[","<","(leq
     )"]],
  [["$\sum \int \ \frac{dy}{dx}$","$[\phantom{aa}]_\cdot^\cdot \ \ \
     frac{\partial z}{\partial x}$"],["$\int$","$\sum$","$[\phantom{a
     }]_\cdot^\cdot$","$\frac{dy}{dx}$","$\frac{\partial z}{\partial x
     }$"],["int(?,,,)","sum(?,,)","br(?,,)","diff(?,)","par(?,)"]],
  [["7","pqrs"],["7","p","q","r","s"],["7","p","q","r","s"]],
  [["8","tuv ="],["8","t","u","v","="],["8","t","u","v","="]],
  [["9","wxyz"],["9","w","x","y","z"],["9","w","x","y","z"]],
  [[") >",""],[")","}","]",">","$\geq$"],[")","\}","]",">","(geq
     )"]],
  [["lim mat $\cdot$","det case $\cdots$"],["lim","mat","det","case","
     $\cdot$"],["lim(?,,)","mat(?,;,)","det(?,;,)","case(?,;,)","dot
     ()"]],
  [["' $\pm$ text","space"],["$\pm$","$\mp$","'","text","space"],["(pm
     )","(mp)","'","tx(?)","(sp)"]],
  [["0","$+ \ - \ \times \ \div$"],["0","-","+","$\times$","$\div$
     "],["0","-","+","\times ","\div"]],
  [["$\pi\ \infty$ !","| "],["$\pi$","$\infty$","!","| ","\"],["pi","(
     inf)","!","| ","\"]],
  [[",","; . : /"],[",",";",":",".","/"],[",",";",":",".","/"]],
  [["",""],["","","","","",""],["","","","","",""]]
];
```

最後に，入力方法以外で kettaskorgv.html を変更したところについて記す.

KeytableW：

キーに複数行表示するために，Keytable を少し変えて，KeytableW というコマンドを作った.

確認ボタン：

学生番号を入力した後に押す 「OK」 ボタンについて，表示を「確認」 とし，2 回押すと「次へ」という表示に変わり，続けて押すことで次の問題に移れるものにした．また,「リセット」ボタンを押したときに「キャンセル」と表示を変え，リセットをキャンセルすることができるようにした．

Rec ボタン：

Rec ボタンを押すと入力したものなどの情報が枠内に表示され，それをコピーして送信する仕様になっていたが，全てを選択しないでコピーして送信してしまうということがときどきあった．そこで，Rec ボタンを押したときに，以下のコマンドを実行することで，表示される内容をクリップボードにコピーするようにした．なお，CindyScript のコマンド javascript() は，cdy.evokeCS() の反対のものであり，CindyScript により

javascript()

と書くことで，() 内のコマンドを JavaScript として実行することができる．

ソースコード 10: クリップボードにコピー

```
1 javatext=replace(rec,"\","\\")
2 javascript("navigator.clipboard.writeText("+Dq+javatext+Dq+");")
```

4 KeT-LMS での入力方法の両立

4.1 入力方法両立のきっかけ

KeT-LMS での入力方法を決めるため，大学 3 年の学生 9 名に実際の画面を使わせて，どれが最も使いやすいかのアンケートを行ったところ，下記の結果となった．

- フリック入力の方が使いやすい：5 名（55.6%）
- キーボードタイプの入力の方が使いやすい：4 名（44.4%）

我々は「学生はフリック入力に慣れており，フリック入力を選択する学生が大半を占めるのではないか」と予想していたが，意外にも学生の好みは拮抗しており，キーボードタイプの入力も無視することができないことがわかった．そこで，フリック入力，キーボードタイプの入力の両方を準備し，実際に KeT-LMS を使ってもらいながらどういった場面についてどちらの入力方法が選ばれているかを調査することになった．

4.2　入力方法の切り替え

入力方法の切り替えは，画面上部の「flick」ボタンと「keyboard」ボタンをクリックすることにより行える.「flick」のボタンをクリックすると 図 1(右) のような画面になり，flick 入力が行えるようになる.「keyboard」のボタンをクリックすると 図 1(左) のような画面になる（これが KeT-LMS のデフォールトの画面である）.

5　システムの概要

5.1　システムの全体構成

本研究で開発されたシステムでは，HTML5 を活用した二層スクリーン構造を導入している．このシステムは，既存のウェブ教材やウェブサイトの上にインタラクティブな層を重ねることで，教材の拡張性と教育効果を高めることを可能にしている．具体的には，下層であるボトム画面が基本的な教材内容を表示し，その上層であるトップ画面には追加のインタラクティブ要素を配置することができる．

5.1.1　ボトム画面の構成

ボトム画面は，主に既存の教材やウェブサイトを表示する役割を担う．HTML5 の `<iframe>` タグを使用して，外部からのコンテンツを埋め込み，表示することが可能である．この層は，教材の基本的な情報提供の場となり，ユーザーに基本的な情報を提供するコンテンツ層である．

5.1.2　トップ画面の構成

トップ画面は，下画面に重ねる形で設置される追加のインタラクティブ層である．この層には，JavaScript と CSS を用いて動的に生成されるボタンやテキストボックスが含まれる．これらの要素は，教材の理解を深めるためのクイズ，注釈の追加，または追加的な情報へのリンク提供など，学習者の参加を促すために設計されている．このトップ画面の要素は，既存の教材の内容に影響を与えることなく，動的に表示・非表示を切り替えることができるため，教材の柔軟な拡張が可能である．

5.1.3　二層スクリーン構造の構成

本システムの二層スクリーン構造は HTML 要素の position プロパティを用いて実現されている．position プロパティは static, relative, absolute, fixed, sticky の 4 つの値を取りうる．それぞれの意味は下記の通りである．

static デフォルトの値であり，要素が通常の文書フローに従って配置される．top, right, bottom, left, z-index の設定が無視される．

relative 要素を通常の文書フローに従って配置し，その位置を基準として，top, right, bottom, left プロパティで指定された値に基づいて位置を調整する．他の要素に対する影響はない．

absolute 要素を通常の文書フローから取り除き，その位置を親要素（position: relative, position: absolute，または position: fixed が設定された要素）に対して配置する．親要素が存在しない場合は，初めの position: relative が設定された祖先要素，または文書の初期コンテナ（root）に対して配置される．top, right, bottom, left プロパティで位置を指定できる．

fixed 要素をビューポート（ブラウザの表示領域）に対して固定位置に配置する．スクロールしても要素の位置が変わらないため，ページの上部や固定されたサイドバーなどに利用される．top, right, bottom, left プロパティで位置を指定できる．

sticky 要素がスクロールされる範囲内で通常の文書フローに従って配置されるが，特定のしきい値を超えたときに固定位置に変わる．通常は親要素に対して設定され，親要素の境界を超える前に固定される．top, right, bottom, left プロパティで固定位置を指定できる．

上のうち，このシステムで重要なのは absolute である．これを用いることで HTML 要素の上に他の HTML 要素を貼り付ける事が可能となり，この仕組みを活用して画面の二層スクリーン構造を実現している．具体的には，本システムの HTML ファイルの二層スクリーン構造に関わる部分は下記のようになっている．

ソースコード 11: HTML ファイルの二層スクリーン構造に関わる部分

```
1 <div id="top_display" style="display: flex; position: relative;">
2   <div id="top_scrren" style="top: 0px; left: 0; margin: auto; position
        : absolute; pointer-events: auto;">
3         ここにトップスクリーン上の HTML 要素を置く
4   </div>
5   <div id="bottom_screen">
6     <iframe id="iframe1" src="kettaskv001F0319a.html" target="_blank
          width="670" height="2000" style="pointer-events: auto;"></
          iframe>
7   </div>
8 </div>
```

id が top_display となっている div 要素が 2 層のスクリーンをまとめている所である．その中にある id が top_screen となっている div がトップ層に対応する．上に書かれているように，この div 要素の下にトップ層の HTML 要素を置く．トップ層に置かれた HTML 要素は x, y 座標の値を left, top のプロパティで指定することにより自由な位置に配置できる．また，id が buttom_screen となっている div がボトム層に対応している．上の例ではここに iframe で KeT-LMS の HTML ファイル kettaskv001F0319a.html を埋め込んでいる．

このシステムの編集モード（教材を作成する際のモード）では図 2 にあるようにトップ層に置かれた HTML 要素の周りにいくつかの小さなボタンが配置され，このボタンを操作

することにより HTML 要素の位置を変えたり，サイズを変更したり，消去したりすることが可能になっている（図 2 の「完了」「Plot」と書かれたボタンや中央右にある Cinderella の図がトップ層に置かれた HTML 要素である）．

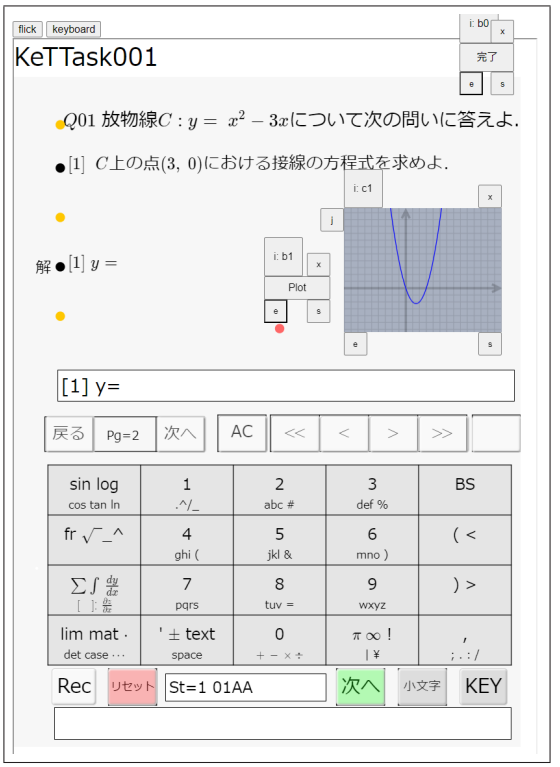


図 2：編集モードの画面

5.1.4　インタラクティブ要素の統合

トップ画面に配置されるインタラクティブ要素は，HTML5，CSS3，および JavaScript を駆使して開発されており，ユーザーからの入力を受け取り，即座に教材の表示を更新するインタラクションが可能である．例えば，テキストボックスを通じてユーザーが入力した回答が正しいかどうかを即座に評価し，フィードバックを提供することができる．また，ボタンを用いて，追加の説明や補足情報を画面に表示することも可能である．これらのインタラクティブ要素を活用した教材の機能については後述する．

5.1.5　教材の機能

上記の二層スクリーン構造を活用することにより，本教材では下記のような機能を実現している．

(a) キーボード⇔フリック入力の切り替え

(b) 使用中の入力方法の種類を 1 秒毎に記録

(c) 問題に応じた絵の表示

(d) 問題に応じた Cinderella の図の表示

(e) 入力に応じた Cinderella の図の変更

(f) 画面の横幅の調整

(g) 演習問題の結果をサーバーへ送信する

以下これらについて解説する.

5.2 (a) キーボード⇔フリック入力の切り替え

入力方法の切り替えは，上記のボトム画面の `iframe` タグの `src` を切り変えることによって実現する．具体的には画面上の「flick」ボタンと「keyboard」ボタンをクリックすることにより，HTML5 の `<iframe>` タグの src をフリック入力形式の KeT-LMS の HTML ファイル (「flick」ボタンをクリックした場合）やキーボード入力形式の KeT-LMS の HTML ファイル (「keyboard」ボタンをクリックした場合）に設定する.

5.3 (b) 使用中の入力方法の種類を１秒毎に記録

先に述べた理由により，教材の利用者がフリック入力，キーボード入力のどちらを使用しているかを調査する必要があるので，下記のコードで使用中の入力方法の種類を１秒毎に記録し，保存するようにしている.

ソースコード 12: １秒毎に記録する

```
1 const intervalId = setInterval(() =>{
2     takelog();
3     if(timecount > timeout){
4       clearInterval(intervalId);
5     }},
6 1000);
```

上の `takelog()` は次のように定義される１秒毎に起動される関数である．実行すると id が log であるテキストアリアにそのとき使用中の入力方法の種類を書き出すようになっている.

ソースコード 13: １秒毎に起動される関数

```
1 function takelog() {
2   let d3 = document.getElementById('test_iframe').contentWindow.cdy.
        evalcs("nqu").value.real;
3   if (mode=="flick") {
4     let d1 = document.getElementById('test_iframe').contentWindow.cdy.
          evalcs("startposf").value.real;
5     if (d1 != 0) {
```

```
6       let d2 = $("#log").val();
7       let d4 = document.getElementById('test_iframe').contentWindow.cdy.
            evalcs("hyoujif").value.real;
8       $("#log").val(d2+"\n"+d3+","+d4);
9       timecount = timecount+1;
10    }
11  } else {
12    let d2 = $("#log").val();
13    let d4 = -1;
14    $("#log").val(d2+"\n"+d3+","+d4);
15    timecount = timecount+1;
16  }
17  for (let key in show_hide) {
18    if (show_hide[key].indexOf(d3) >= 0) {
19       $(key).show();
20    } else {
21       $(key).hide();
22    }
23  }
24 }
```

5.4 (c) 問題に応じた絵の表示

本システムでは，トップ画面に埋め込んだ絵を問題番号に応じて，表示したり，消したりできるようになっている．この機能は特定の問題へのヒントの表示などへ使うことを想定している．どの絵をいつ表示するかは JavaScript の連想配列の形式で指定するようになっており，例えば

```
{ "#c1":[2,3], "#f1":[1]}
```

と指定されたときは，id が c1 の図を問題番号が 2,3 の時に表示し，id が f1 の図を問題番号が 1 のときに表示するというようになっている．

具体的に図を表示したり，消したりするのは上の関数 `takelog()` が行っており，

ソースコード 14: 図を表示・消去する

```
1   for (let key in show_hide) {
2     if (show_hide[key].indexOf(d3) >= 0) {
3       $(key).show();
4     } else {
5       $(key).hide();
6     }
7   }
```

の部分がそれに相当する．

5.5 (d) 問題に応じた Cinderella の図の表示

本システムでは，トップ画面に埋め込んだ Cinderella の図も絵と同様に問題番号に応じて表示したり，消したりできるようになっている．Cinderella の図をいつ表示するかは絵と同様に JavaScript の連想配列の形式で指定するようになっている．

5.6 (e) 入力に応じた Cinderella の図の変更

本システムでは，埋め込んだ Cinderella の図を KeT-LMS への入力に応じて変更することが可能になっている．例えば，Cinderella の図に $y = x^2 - 3x$ と KeT-LMS へ入力した関数のグラフを同時に描画するには，次の JavaScript のコードを実行する．

ソースコード 15: 関数を描画する JavaScript のコード

```
1 var fx = get_ketmath_input_raw();
2 var com = "plot(x*x-3*x); plot("+fx+")";
3 csdraw('c1',com);
```

1行目の `get_ketmath_input_raw()` は KeT-LMS への入力を取り出す関数である．2行目で関数を描画する Cinderella の命令を `com` という変数に代入する．3行目の `csdraw('c1',com)` は `c1` という id の付いた Cinderella に `com` を draw の命令として実行させる命令である．

1行目の `get_ketmath_input_raw()` は次のように定義されている．

ソースコード 16: get_ketmath_input_raw() のコード

```
1 function get_ketmath_input_raw() {
2   return(document.getElementById('test_iframe').contentWindow.cdy.
3             evalcs('Text2').value.text);
4 }
```

本システムでは KeT-LMS は `test_iframe` という id の `iframe` に埋め込まれており，上の関数 `get_ketmath_input_raw()` は埋め込まれた html ファイル上で

```
cdy.evalcs('Text2').value.text
```

という命令を実行してその結果を返す，という内容になっている．KeT-LMS の入力のテキストボックスは，Cinderella 上では Text2 という名前になっており，上記の命令はその中に入力された文字列を返す JavaScript の命令である．

3行目の `csdraw()` は次のように定義された JavaScript の関数である．

ソースコード 17: csdraw() のコード

```
1 function csdraw(cindyname,com) {
2   csact("draw",cindyname,com);
3 }
4 function csact(act,cindyname,com) {
5   let idx = Number($("#"+cindyname).attr("index"));
6   $("#cs"+idx+act).html(com);
7   eval(Palette_Cindy.cinderella_code[idx]);
8 }
```

Cinderella を HTML ファイルに変換すると，初期化の命令は

```
<script id="csinit" type="text/x-cindyscript"></script>
```

というタグに埋め込まれ，draw の命令は id を csdraw とする同様のタグに埋め込まれる．このシステムでは，Cinderella の図を複数埋め込むことを可能にするために埋め込まれた Cinderella の図に番号を付け，タグの id にその番号を付加している．例えば，１番目の Cinderella の図に対する初期化の命令は

```
<script id="cs1init" type="text/x-cindyscript"></script>
```

というタグに埋め込まれ，draw の命令は

```
<script id="cs1draw" type="text/x-cindyscript"></script>
```

というタグに埋め込まれている．

ソースコード 17 の関数 `csact()` は変数 `act` (`draw` や `init` の値を取る) と `cindyname` (Cinderella を埋め込んだときの id が代入された変数) から上記の `cs1init` や `cs1draw` などのタグの id を生成し，そのタグを持つ div 要素の中身を変数 `com` の値で置き換える．その後，`cindyname` の id を持つ Cinderella を再起動することで `cs1init` や `cs1draw` などタグに書かれた内容を Cinderella に読み込ませる．ソースコード 17 の関数 `csdraw()` はこの `csact()` において変数 `act` の値を `draw` と置くことで，`cindyname` の id を持つ Cinderella の `draw` のアクションを変更している．以上をまとめると，次のようにすることで Cinderella の入力窓への入力された命令を元に Cinderella の図の変更を行っている（以下では埋め込まれた Cinderella の id は `c1` であり，その図に割り当てられた番号は 1 であるとする）．

(1) `get_ketmath_input_raw()` を用いて，Cinderella の入力窓への入力された文字列を取り出す．

(2) 取り出された文字列をもとに，Cinderella の draw に対する命令を作る．

(3) 関数 `csdraw('c1',` (2) で生成した draw に対する命令 `)` を呼び出す．

(4) 関数 `csact()` が呼び出され，`cs1draw` の id をもつタグの内容を「(2) で生成した draw に対する命令」で書き換え，番号が 1 の Cinderella を再起動し，タグに書かれた命令を読み込む．

5.7 (f) 画面の横幅の調整

本システムでは使われている画面の横幅を調べ，埋め込まれた KeT-LMS がその画面の横幅にぴったり一致するように KeT-LMS の拡大縮小を行っている．このように画面サイズを自動調整することにより，学生が自分でサイズを調整する必要がなくなった．

画面サイズの調整を行っているコードを示す（コードの中の変数 `epagew` は KeT-LMS の横幅を表している）．

ソースコード 18: 画面サイズの調整

```
function set_scale(selector,scale) {
 $(selector).css({ 'transform-origin' : '0 0','transform' : 'scale('+
     scale+')'});
}
set_scale("#scale_div",(document.documentElement.clientWidth)/epagew);
```

5.8 (g) 演習問題の結果をサーバーへ送信する

本システムでは，課題を解き終わったあと，ボタンを１つクリックするだけでその解答とログのデータをサーバーに送信するようになっている（KeT-LMS では，ボタンをクリックした後，テキストボックスに表示される文字列を学生が自分で送信する必要があり，文字列の一部がうまく送信されないなどのトラブルがたまに起きていた）．この機能は，KeT-LMS との整合性を考えて画面の右上に新しいボタンを貼り付けて実現している（図 2 の右上にある「完了」のボタン）。第５節に述べた二層スクリーン構造を本システムが採用し，下層に KeT-LMS，上層にインタラクティブ層を配置することで，埋め込まれた KeT-LMS には手を加えることなく課題提出のボタンの機能変更が可能になっている．

実際にサーバーに解答結果とログのデータを送信する関数 check_submit_data() のコードをいかに示す．

ソースコード 19: サーバーに解答結果とログのデータを送信する

```
function check_submit_data() {
  let con = confirm("問題の解答とキー入力のログをサーバーに送り,課題を終
      了しますが,それで良いですか? ");
  if (!(con)) {
    return(0);
  }
  document.getElementById('test_iframe').contentWindow.cdy.evokeCS('rec=
      rechead+Getcurtime()+";;"; ansL_nqu=Textedit(ch,"","");  str="";
      forall(1..(length(ansL)),n,str=str+ansL_n+";;";); rec=rec+
      substring(str,0,length(str)-2); Subsedit(9,rec); StrL_4=rec;');
  document.getElementById('test_iframe').contentWindow.cdy.evokeCS('
      jsrec = replace(rec,"\'","[[dash]]")');
  let ansdata = document.getElementById('test_iframe').contentWindow.cdy
      .evalcs('jsrec').value.replaceAll("[[dash]]","'") + "\n";
  let logname = document.querySelector("#logname_data_1").value;
  let fname = "KeT-LMS/"+ logname + ".log";
  let uname = ansdata.split(";;")[0];
  let funame = "KeT-LMS/" + logname + "_" + uname + ".klog";
  let ans = file_check(funame);
  if (ans == "TRUE") {
    alert("課題は既に提出されています");
  } else {
    submit_data(ansdata,logname,fname,funame);
  }
```

```
19 }
```

上のソースコードで `file_check(funame)` は `funame` という名前のファイルがすでにサーバーにある場合は TRUE を，そうでない場合は FALSE を返す関数である．また，`submit_data()` は解答とログのデータをサーバーに送信する関数である（サーバー側では PHP のプログラムがデータを受け取り，サーバーのファイルに書き込むようになっている）．

6　教材の作成例

これまでに紹介した機能を活用して作成した E-Learning 教材の例を紹介する．この E-Learning 教材にアクセスすると，図 3 のような画面になる（左図，中央図，右図はそれぞれ「起動直後の画面」,「画面のどこかをクリックまたはタッチした後の画面」,「 keyboard のボタンをクリックした後の画面」である）．なお，上の教材の機能「(f) 画面の横幅の調整」で解説したように，本教材は画面の拡大縮小を自動的に行い，教材の横幅が IT デバイスの横幅にちょうど合うようになっている．上記のように本教材では，フリック入力と

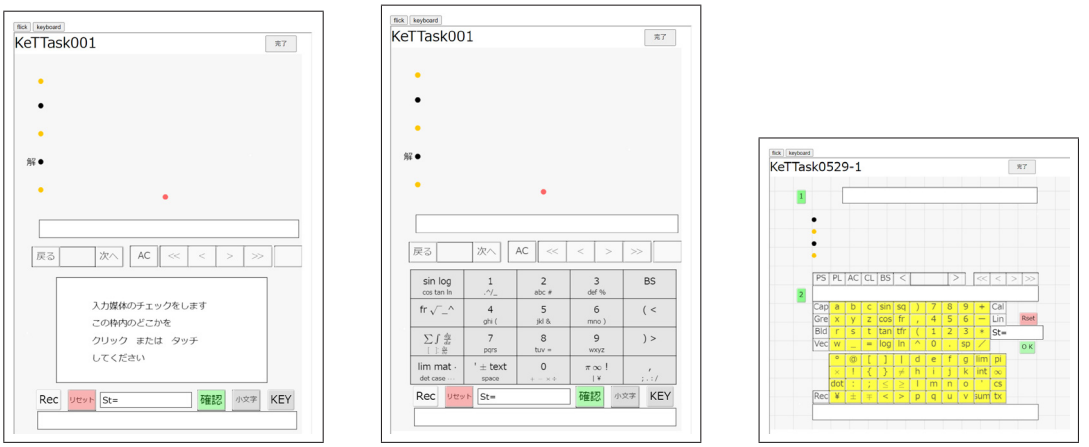


図 3：起動時の画面

キーボード入力のどちらでも好きな方を選択できるようになっているが，説明の簡単のためここではユーザーがフリック入力を選択したと仮定して，以下の解説を行っていく．

この起動時の画面で画面の下（フリック入力の場合），または右中央（キーボード入力の場合）の「 St= 」の右に生徒番号を入力して，進めると図 4 の左のような画面になり，問題文が表示される．あとは「次へ」と「戻る」のボタンを使って次の問題へ行ったり，もとに戻ったりすることができる．例えば，図 4 の左の画面で「次へ」のボタンをクリックすると，図 4 の右の画面のようになり，画面中央のテキストボックスに ”[1] y= ”と表示され，接線の方程式の入力が促される．図 4 の右の画面にはそれまで表示されていなかった Cinderella の図と「Plot」と書かれたボタンが表示されているが，これは上の教材の機能「(d) 問題に応じた Cinderella の図の表示」の所で述べた機能を用いており，問題文「C 上の点 $(3, 0)$ における接線の方程式を求めよ.」が出題されたときに表示されるようになっている．

図 4：問題文の表示の画面

また，この画面で ”[1] y= ”の右に接線の式を入力し，画面にある「Plot」のボタンをクリックするとその関数のグラフを Cinderella の図の中にプロットする．これは教材の機能「(e) 入力に応じた Cinderella の図の変更」を用いており，例えば，” y= ” の右に 3x-9 と入力して「Plot」のボタンをクリックすると図 5 の左のような画面になる．この図を見てユーザーは入力した接線の方程式 y = 3x-9 が正しく $(x, y) = (3, 0)$ での接線となっていることを確認できる（または間違った接線の式を入力してしまった場合はその間違いに気づくことができる）．

ここで「次へ」をクリックすると，今後は次の [2] の問題「C 上の点 $(1, -2)$ における接線の方程式を求めよ.」が表示される．この教材では，この問題に対しても Cinderella の図と「Plot」のボタンが表示されるように設定しており，生徒が自分の入力した接線の方程式が正しいかどうかを確認できるようにしている．$(x, y) = (1, -2)$ での接線の方程式 y = -x-1 を正しく入力して,「Plot」をクリックした場合の画面を図 5 の右に示す．この図 5 の右の画面には，それまでにない図（画面の上部左にある数式が書かれたもの）が表示されているが，これは教材の機能「(c) 問題に応じた絵の表示」を用いて表示させたものである．このように問題に応じたヒントを画面に表示させることも可能である．

問題を解き終わったら，最後に画面の右上にある「完了」のボタンをクリックすると生徒が入力した問題の解答と生徒がどの入力システムを用いたかのログデータがサーバーに送信される（これは教材の機能「(g) 演習問題の結果をサーバーへ送信する」を用いている）．この「完了」のボタンは KeT-LMS のボタンではなく，その上のトップ画面に貼り付けられた JavaScript のプログラムを起動するボタンである．このように KeT-LMS のプログラムを変更することなくその上に新たなボタンを貼り付けて新しい機能を追加することが可能である．この「完了」のボタンは用いるにはサーバーが必要になるので，サーバーが準備できない場合は画面の左下にある「Rec」のボタンを使えば良い（「Rec」のボ

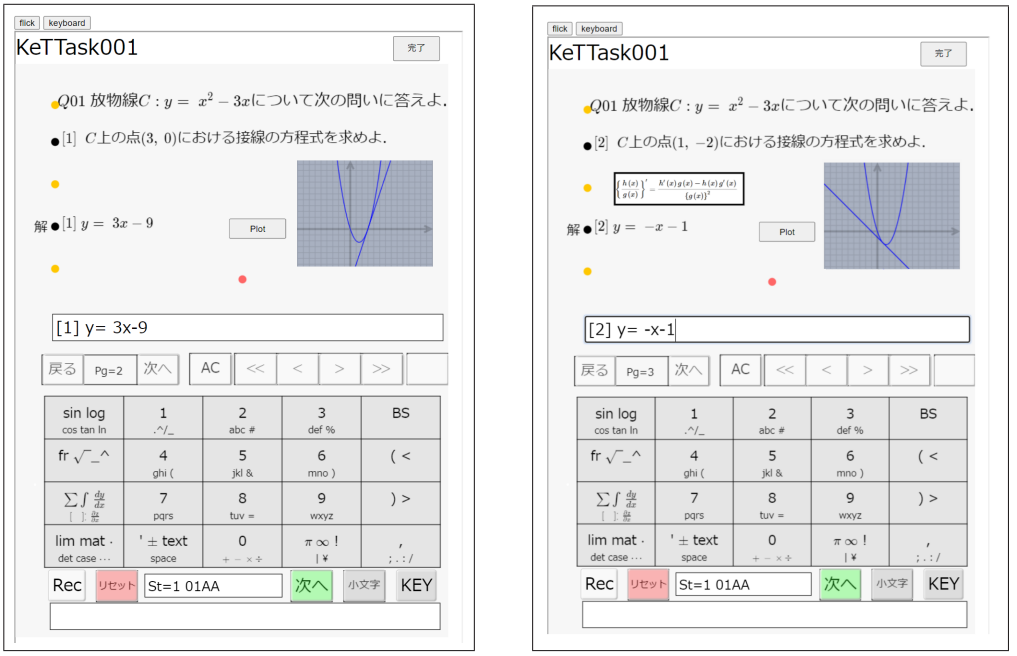


図 5：ユーザーからの入力に応じて，Cinderella の図を変更する

タンは第 3 節で説明したように, ユーザーがこれまでに入力した解答をクリップボードにコピーする命令である）.

7　まとめと今後の課題

本研究では，KeT-LMS におけるスマートフォン利用の効率を向上させるためにフリック入力機能の導入について検討し，その可能性について検討した．本論文では特にその技術的な側面に焦点を当て，フリック入力の機能の実現，フリック入力とキーボード入力の両立やその他のユーザーに取って有用であると思われる機能について解説を行った．具体的には，フリック入力とキーボード入力をユーザーが簡単に切り替えられる機能，使用状況を記録するログ機能の実装が行われ，これらが今後のシステム改善に役立てられる見通しである．

従来のキーボード入力に比べて，特に画面の狭いデバイスでの入力ミスが減少し，ユーザーの利便性が大幅に向上することが期待されるが，実際に上記のログ機能を用いた実験を行い，その結果を調査することでユーザーの視点からの検証を行うことが今後の課題である．それを含めた今後の課題を以下に挙げる．

1. **ユーザーエクスペリエンスの評価と改善**：上で述べたように，フリック入力機能の導入によってユーザーの利便性が向上したかどうかを定量的に評価する必要がある．前述したように本システムではユーザーがどの入力方法を用いたかのログを記録しており，そのデータの解析を行う．

2. **実証実験と広範な導入**：フリック入力機能の効果を検証するために，実際の教育現場での実証実験を行い，その成果を基にシステムの改良を重ねることが重要である．

3. **拡張機能の開発**：フリック入力の導入に加え，さらに学習体験を向上させるための新機能の可能性を検討する．例えば，音声入力や手書き入力の導入，AI を活用した学習支援機能の追加などが考えられる．

4. **セキュリティとプライバシーの強化**：学習データや個人情報の保護は極めて重要であり，データの暗号化やアクセス制御の強化など，セキュリティ対策を徹底する必要がある．また，ログ機能によって収集されたデータのプライバシー保護についても慎重な検討を要する．

参考文献

[1] 高遠節夫, 濱口直樹, 北本卓也, 1 次元表現ルールに基づいた数式の送受と授業実践, 城西大学数学科数学教育紀要 **4** (2023), 23–34.

[2] 高遠節夫, 碓氷久, 西浦孝治, 濱口直樹, KeT-LMS の開発と授業実践, 城西大学数学科数学教育紀要 **5** (2024), 38–49.

[3] KeTCindy Home, https://s-takato.github.io/ketcindyorg/indexj.html