# メロディ付き物語創作支援 Web アプリケーションの開発

力武研究室　情報工学科 5 年　山下峻

指導教員　情報システム工学科　力武克彰

# 目次

# 1. はじめに

## 1.1. 研究背景

### 1.1.1. メロディ付き物語創作支援システム

児童が楽しみながら表現力や創造力を養えるソフトウェアとして，「メロディ付き物語創作支援システム（muphic)」が亀谷学人氏，瀬戸敏文氏らによって開発された[1].

muphic は物語（絵）の作成と作曲を行えるソフトウェアである. 児童はまず物語を作り，次にその物語の雰囲気やストーリーに合うようなメロディを考えて作曲する．この 2 つの手順を通して，児童は表現力などを養うことが出来る.

物語の作成は，図 1.1 に示すような ユーザインタフェースで行う.


**図 1.1** 物語作成画面

児童は物語の「背景」を選択し，その上に「人物」「動物」「道具」の 3 種類のイラストをスタンプのように配置して 1 枚の絵を作る．この絵には文章をつけることも出来，物語のストーリーを文字で表現することも出来る. このような絵を複数作り，紙芝居のような構成とすることで一つの物語を形成する.

作曲は，図 1.2 に示すようなユーザインタフェースで行う．


**図 1.2** 作曲画面

作曲を行うにあたって，児童はピアノやオカリナなどの様々な種類の楽器を選ぶことが可能である．楽器は動物として表現されており，画面右側に動物の絵柄で楽器のボタンが並んでいる．ボタンをクリックして楽器を選択すると，画面中央の譜面上に音符を配置することが出来る．譜面は児童にとって難解な五線譜による表現ではなく，視覚的にも理解しやすいシリンダーオルゴールのような表現がなされている．作曲したメロディは再生して試聴することも可能であり，メロディの五線譜出力も可能である．

物語とメロディを作成したら，それら 2 つを合わせて同時に再生・鑑賞することが出来る．同時に再生しながら「作った絵に合うようなメロディはどうすれば作れるか」「作曲したメロディに合うような絵はどのような雰囲気のものか」と試行錯誤を繰り返すことにより，児童は表現力や創造力を養うことが出来る．

### 1.1.2. 出前授業による評価

muphic は実際に使用可能なシステムとして完成しており，muphic を用いた出前授業が既に何度か実施されてきた．出前授業は小学校や児童館を対象としたものであり，実際に児童に muphic を遊んでもらい，作品を作ってもらうという授業が展開された．授業の終了時には児童に対してアンケートを取り，muphic の使い心地などの調査を行った．その結果として，児童から「楽しかった」「思い通りの作品が作れた」「音色や音階の違いがわかった」「また遊びたい」といった肯定的な意見を数多く得ることが出来た[2]．

以上の結果から，muphic は児童が楽しみながら情操を養えるツールとして有用であることが立証された．

### 1.1.3. muphic の Web アプリケーション化

muphic は児童の情操教育ソフトウェアとして有用であることが確認されたが，いくつかの問題点により，小学校での運用はまだ困難であることも明らかとなった．まず，小学校のコンピュータ教室で muphic を使用するとなると，数十台もの PC 端末へ muphic をインストールする必要がある．このインストール作業を担当教員が 1 人で行う場合，手間と時間を要してしまい導入が容易でないことが問題である．また，導入を完了したとしても，コンピュータ教室などのごく限られた教室（空間）での使用，端末台数や実習時間の不足など数多くの問題が残されていた．

このような問題を解決する手段の 1 つとして，muphic の Web アプリケーション化が提案された[3]．Web アプリケーションは Web ブラウザからアクセスして使用するシステムである．よって，小学校の PC 端末に Internet Explorer などの Web ブラウザさえ入っていれば，ソフトウェアのインストールを行うことなく muphic を遊ぶことが出来る．また，Web アプリケーションはどこからでもアクセス出来るようなオンラインアプリケーションに分類される．つまり，小学校だけでなく自宅などからでも自由にアクセスし，muphic を遊ぶことが可能となる．自宅でならば，児童は端末台数や授業時間などを気にすることなく，満足するまで muphic を楽しむことが出来る．

## 1.2. 研究の目的と概要

先行研究では，児童向け情操教育ソフトウェアの有用性が立証されたものの，その運用はまだ困難であることが確認された．その解決策として Web アプリケーションでの実装が提案された．

本研究では，児童が楽しみながら創造力や表現力を養うことが出来る Web アプリケーションとして，「メロディ付き物語創作支援 Web アプリケーション（muphic-online）」を開発することを目的とする．

本研究では以下に示す流れで muphic-online の開発を行った．

I. システムの要求分析

muphic-online を開発するにあたり，最初にシステムの要求分析を行った．要求分析ではユーザ（児童）の視点でシステムへの要求を洗い出した．

II. システムの仕様決定

要求分析で洗い出した要求を基に，muphic-online の仕様を決定した．

muphic-online のシステム構成を，ユーザインタフェースの機能をまとめたクライアント面と，データ管理機能をまとめたサーバ面とに分割し，それぞれについての仕様をまとめた．クライアント面ではログイン，物語作成，作曲の 3 種類の操作の定義を行い，それらを行うための画面を定義した．サーバ面ではデータの読み込みと保存を可能とするためのシステムアーキテクチャの構想を定義した．

III. システムの実装手段の検討・決定

クライアント面とサーバ面について，それぞれの仕様を実現するために必要な機能を洗い出し，それらの機能の実装手段（プログラミング言語など）を検討し，決定した．

IV. システムの実装

採用した実装手段を用いて muphic-online を実装した．

クライアント面の実装として，ログイン，物語作成，作曲を行うための画面をそれぞれ実装した．サーバ面の実装として，データの読み込みと保存を可能とするシステムアーキテクチャを設計し，実現した．

V. システムに対する考察

実装した muphic-online について，要求と仕様を満たすことが出来たか考察した．結果として，要求と仕様を概ね満たしたシステムを開発することが出来た．

結論として，開発した muphic-online はユーザの要求を概ね満たしているものの，機能の充実度や利便性の観点からまだ運用に堪えうるものではないことを示す．そして，運用可能なシステムとするために，より一層の機能の充実や利便性の向上を行っていく必要性があることを述べる．

### 1.3. 論文の構成

本論文は本章を含めて 10 章から構成される．

第 1 章では，研究背景と研究目的および概要について述べた．

第 2 章では，muphic-online の開発にあたって最初に行った要求分析について述べ，洗い出した要求をまとめる．

第 3 章では，muphic-online のシステム構成の分割について述べ，クライアント面とサーバ面のそれぞれについて要求を基に決定した仕様をまとめる．

第 4 章では，muphic-online の仕様を実現するために必要な機能を，クライアント面とサーバ面のそれぞれについてまとめる．

第 5 章では，muphic-online の実装に採用した実装手段の詳細を，クライアント面とサーバ面のそれぞれについて述べる．

第 6 章では，第 5 章で述べた実装手段について改めてまとめる．

第 7 章では，muphic-online の実装結果を，クライアント面とサーバ面のそれぞれについて示す．

第 8 章では，実装した muphic-online がクライアント面とサーバ面のそれぞれの仕様を満たし，ユーザの要求を満たしているか考察する．また，実装の工夫点についても述べる．

第 9 章では，開発した muphic-online について，今後行うべきシステムの評価や利用のされ方を述べる．

第 10 章では，本研究のまとめと結論を述べる．

## 2. システムの要求分析

本章では，システムの開発にあたって最初に行った要求分析についてまとめる．要求分析に関しては先行研究の muphic の要求分析内容も参考にし，児童でも容易に使用可能なシステムを目指した．

### 2.1. 要求分析方法

要求分析はユーザストーリーを作成して行った．ユーザストーリーとは，ユーザがシステムに対して求める機能をユーザ視点で記述したものである．一般的には「（ユーザ）として（機能）をしたい．それは（理由）だからだ．」という形式で記述される[4]．ユーザはシステムを使用するユーザの種類であり，アクターである．機能は文字通りシステム内の機能であり，ユースケースである．ユーザストーリーは「誰が，どうして，何をしたいのか」という重要な疑問に対する答えを示すことにより，明確な要求分析を行う助けとなる．

### 2.2. 用語の定義

muphic-online のユーザストーリー導出にあたって，muphic で用いられているオブジェクトや概念を新たな用語として定義し直し，それらの用語をユーザストーリーの記述に使用した．それらの用語について以下にまとめる．

- 物語に関する用語
  - 背景イラスト
    物語の背景を彩る風景のイラスト画像を示す．
  - 人物イラスト
    物語に使用出来る人のイラスト画像を示す．
  - 動物イラスト
    物語に使用出来る動物のイラスト画像を示す．
  - 道具イラスト
    物語に使用出来る道具のイラスト画像を示す．
  - プレビュー画面
    背景イラスト，人物イラスト，動物イラスト，道具イラストを配置して 1 枚の絵を作成するためのスペースを示す．
  - 消すボタン
    プレビュー画面上に配置した人物イラスト，動物イラスト，道具イラストを削除するためのボタンを示す．

- 作曲に関する用語
  - 音符

    動物で表現された，メロディを形作るためのオブジェクトを示す．音符には音色の情報が含まれる．
  - 譜面

    音符を配置してメロディを作成するためのスペースを示す．
  - 消すボタン

    譜面上に配置した音符を削除するためのボタンを示す．

### 2.3. ユーザストーリー

要求分析では，まずユーザストーリーを全て導出し，各ユーザストーリーを機能の内容に着目して分類した．対象ユーザは児童（小学校低学年～中学年）で統一としたため，各ユーザストーリーには機能のみを記述した．以下に，導出したユーザストーリーとその分類をまとめる．また，各ユーザストーリーには「U-（番号)」の形式で番号を付与する．

- アカウント

  U-1　名前を入力してログインがしたい．それは，ユーザ毎に作品を作りたいから

- 物語

  U-2　物語に使用するイラストの一覧を見たい．それは，物語作成に使用出来るイラストをわかりやすく確認したいから

  U-3　物語に使用する背景イラストを選択したい．それは，物語の背景イラストを変えたい時があるから

  U-4　物語に使用する人物イラストを選択したい．それは，物語に使う人物イラストを変えたい時があるから

  U-5　選択した人物イラストをプレビュー画面上に配置したい．それは，選択した人物イラストを使って物語を作りたいから

  U-6　物語に使用する動物イラストを選択したい．それは，物語に使う動物イラストを変えたい時があるから

  U-7　選択した動物イラストをプレビュー画面上に配置したい．それは，選択した動物イラストを使って物語を作りたいから

  U-8　物語に使用する道具イラストを選択したい．それは，物語に使う道具イラストを変えたい時があるから

  U-9　選択した道具イラストをプレビュー画面上に配置したい．それは，選択した道具イラストを使って物語を作りたいから

  U-10　消すボタンを用いてプレビュー画面上の人物イラスト，動物イラスト，道具イ

ラストを削除したい．それは，間違えて作成した物語を修正したいから

U-11　以前保存した物語を読み込みたい．それは，前回までの物語作成を再開したい時があるから

U-12　作成した物語を保存したい．それは，どこからアクセスしても同じデータを使って物語作成がしたいから

U-13　物語を保存した際に，正しく保存が完了出来たか確認したい．それは，自分の物語がきちんと保存されているか気になるから

U-14　物語作成を始める前に，物語の作り方（手順）を見たい．それは，物語を作る時に最初に何をすればいいのか知りたいから

- 作曲

U-15　譜面上に配置する音符を選択したい．それは，作曲に使う音色を変えたい時があるから

U-16　音符を譜面上に配置したい．それは，選択した音符を使ってメロディを作りたいから

U-17　消すボタンを用いて譜面上の音符を削除したい．それは，間違えて作曲したメロディを修正したいから

U-18　譜面のメロディを再生したい．それは，自分で作曲したメロディを試聴して確認したいから

U-19　以前保存したメロディを読み込みたい．それは，前回までの作曲を再開したい時があるから

U-20　作曲したメロディを保存したい．それは，どこからアクセスしても同じデータを使って作曲がしたいから

U-21　メロディを保存した際に，正しく保存が完了出来たか確認したい．それは，自分のメロディがきちんと保存されているか気になるから

U-22　作曲を始める前に，作曲の仕方（手順）を見たい．それは，メロディを作曲する時に最初に何をすればいいのか知りたいから

### 2.4. インタフェース

児童でも簡単に操作が可能なシステムを実現するため，入力にはキーボードを用いずマウスのみを使用することにした．

システムの操作画面であるユーザインタフェースに関しては，児童の興味・関心を惹き，親しみやすいデザインを実現する必要がある．そこで，muphic を参考にし，絵本のようなかわいらしいイラスト画像を効果的に使用した GUI（Graphical User Interface）を実現することにした[5]．ボタンなどにもイラスト画像を使用し，無機質なデザインにならないよう留意する．また，低学年の児童が遊ぶことも考慮し，システムで使われるテキストは基本的

に平仮名で記述することにした.

# 3. システムの仕様

本章では，要求分析の結果を基に定めたシステムの仕様についてまとめる．

## 3.1. システムの構成

システムの仕様を決めるにあたり，システムの全体構成を 2 つに分割した．ユーザが実際に触れる機能（ユーザインタフェース）をまとめたクライアント面と，データの管理機能をまとめたサーバ面とで分けて，それぞれの仕様を定めた．

## 3.2. クライアント面の仕様

クライアント面はユーザインタフェースに関わる機能をまとめたものである．ここでは，muphic-online の主要な機能である物語作成と作曲，ログインに関する仕様と，それぞれの操作を行うための画面の仕様をまとめる．

### 3.2.1. ログインの仕様

muphic-online ではユーザ毎に作品データを保存し，次回遊ぶ際に再び作品データを読み込んで作業を再開出来る機能を実現する．そのためには，ユーザと作品データの紐づけを行う仕組みが必要である．その仕組みとして，muphic-online ではログイン機能を実現する．

ログインの定義を「ユーザ名を入力すること」とする．ログインは単なる文字入力操作であるため，ログアウトは定義しない．入力されたユーザ名を基にユーザを特定し，ユーザと作品データを紐づけて保存と読み込みを実行可能とする．ユーザ名には平仮名，片仮名，アルファベットの文字を使用し，最大 15 文字までの入力を可能とする．

### 3.2.2. ログイン画面の仕様

ログインを行うための画面をログイン画面として定義する．ログイン画面ではユーザ名の入力を行う．

ログイン画面には以下に示すボタンやオブジェクトを配置する．

- 文字ボタン
  キーボードのように文字のラベルがつけられたボタンである．
  ユーザはこのボタンをクリックすることで文字の入力を行う．
  文字ボタンには平仮名，片仮名，アルファベットの 3 種類がある．
- 消すボタン
  入力した文字列を削除するボタンである．
  1 回クリックする毎に，入力された文字列を末尾から 1 文字ずつ削除する．
- テキストボックス
  入力された文字列を表示するスペースである．

1 文字以上の入力がある場合は入力された文字列を表示する．
入力が 1 文字もない場合は，文字ボタンによる入力を促すようなテキストを表示する．

- 名前決定ボタン
  ユーザ名を確定するボタンである．
  このボタンをクリックすることで，その時点で入力されていた文字列がユーザ名として確定され，ログインを完了する．

文字を入力するために文字ボタンを配置することで，マウスによる文字入力を可能とする．文字ボタンによる文字入力はログイン画面における入力であり，テキストボックスによる文字列表示はログイン画面における出力である．

ログイン画面でユーザ名の入力機能と確定機能を実装することにより，ログインに関わる要求である U-1 を満たすことが出来る．

### 3.2.3. 物語作成の仕様

物語作成では，様々な種類の画像を用いて物語（絵）を作る．物語作成に使用出来る画像には背景イラスト，人物イラスト，動物イラスト，道具イラストの 4 種類がある．背景イラスト，人物イラスト，動物イラスト，道具イラストについてはそれぞれ複数種類用意し，ユーザが様々な組み合わせで物語を作ることを可能とする．

### 3.2.4. 物語作成画面の仕様

物語作成を行うための画面を物語作成画面として定義する．物語作成画面では画像を自由に配置して物語の作成を行う．

物語作成画面には以下に示すボタンやオブジェクトを配置する．

- 作曲遷移ボタン
  作曲画面へ遷移するボタンである．
  ユーザがこのボタンをクリックすることで，ブラウザ上に表示される画面が作曲画面に切り換わる．
- 背景ボタン
  物語に使用する背景イラストを選択するためのパレット（後述）を呼び出すボタンである．
- 人物ボタン
  物語に使用する人物イラストを選択するためのパレットを呼び出すボタンである．
- 動物ボタン
  物語に使用する動物イラストを選択するためのパレットを呼び出すボタンである．

- 道具ボタン

  物語に使用する道具イラストを選択するためのパレットを呼び出すボタンである．

- パレット

  背景イラスト，人物イラスト，動物イラスト，道具イラストのそれぞれについて，使用する画像を選択するためのパネルである．

  パレットは背景，人物，動物，道具の 4 種類についてそれぞれ用意し，背景パレット，人物パレット，動物パレット，道具パレットと定義する．各パレットはそれぞれの種類のイラストを全て一覧表示し，その中からクリックされたイラストを選択する機能を提供する．

  各パレットの表示中は画面全体を暗転させ，他のボタン類の操作を不可とする．一覧表示された画像をクリックして選択するか，各パレット内に配置された「閉じるボタン」をクリックすることで各パレットを閉じることが出来，ボタン類の操作も可能となる．

- プレビュー画面

  背景イラスト，人物イラスト，動物イラスト，道具イラストを配置して物語を形成するスペースである．

  初期状態では何も表示されない真っ白な状態である．背景パレットによって背景イラストが選択されると，自動的にプレビュー画面全体の背景がその画像に切り換わる．人物イラスト，動物イラスト，道具イラストのいずれかを選択している状態でプレビュー画面上にマウスポインタを置くと，選択している画像が半透明で表示され，クリックすることでその座標上に画像を配置する．画像は重ねて配置することも可能である．

  人物イラスト，動物イラスト，道具イラストのいずれかを選択中は，プレビュー画面上に配置するまで他のボタン類の操作を不可とする．

- 消すボタン

  プレビュー画面上に配置した人物イラスト，動物イラスト，道具イラストを削除するボタンである．

  クリックすることで消すボタンが押下状態となり，その状態でプレビュー画面上の画像にマウスポインタを置くと，画像上にフォーカスが表示される．クリックすることでその画像を削除する．消すボタンの押下状態はもう一度消すボタンをクリックすることで解除出来る．

- 呼び出すボタン

  前回までの作業で作成した物語のデータを読み込むボタンである．

  クリックすると，前回保存した物語の作品データを読み込み，プレビュー画面内に描画する．その際，現在プレビュー画面内に配置されている画像は全て削除され，読み込んだ作品データでプレビュー画面全体が再描画される．

- 残すボタン

  作成している物語のデータを保存するボタンである．

  クリックすると，現在プレビュー画面内に描画されている背景イラスト，人物イラスト，動物イラスト，道具イラストの全ての画像のデータを，作品データとして保存する．
- 物語保存確認ダイアログ

  物語を保存した際に表示されるダイアログボックスである．

  物語保存確認ダイアログには物語の保存が正常に完了した旨のメッセージを記述し，ユーザにデータの保存完了を報告する．
- 物語導入ダイアログ

  物語作成を開始する際に表示されるダイアログボックスである．

  物語導入ダイアログには物語を作る手順を示し，ユーザが最初に何をすればいいのかを提示する．

物語作成に必要な操作を全てボタンで呼び出すことにより，マウスのみでの操作を実現する．プレビュー画面への画像の配置は物語作成画面における入力であり，プレビュー画面上での画像の表示は物語作成画面における出力である．

物語作成画面では，作曲画面（後述）への遷移も可能とする．これは，「物語の雰囲気に合うようなメロディを作曲する」という作業手順を実現するため，物語作成と作曲を並行して行う必要があるためである．作曲画面へ遷移する際は，物語作成画面で作られた絵のデータはそのままの状態で保持される．

物語作成画面でパレット機能と各パレットを呼び出すボタン，プレビュー画面，消すボタンを実装することにより，基本的な物語作成に関わる要求である U-2～U-10 を満たすことが可能である．そして，呼び出すボタンと残すボタン，物語保存確認ダイアログを実装することにより，物語の読み込みと保存に関わる要求である U-11 ～ U-13 を満たすことが可能である．最後に，物語作成導入ダイアログを実装することにより，物語作成の手順確認に関わる要求である U-14 を満たすことが可能である．

### 3.2.5. 作曲の仕様

作曲では，音符を用いてメロディを作る．広い範囲の音階や豊富な音色，音符の長さなどは音楽の表現の幅を広める反面，作曲で出来ることの範囲を広げすぎてしまい音楽知識を持たない人は寧ろ戸惑ってしまうと考えられる．作曲初心者への配慮と容易な操作を優先し，作曲では以下のように音楽表現の範囲を限定した[6]．

- 使用出来る楽器（音色）は 1 種類とし，ピアノを用いる
- 音階の範囲はド（C4）からド（C5）までの 1 オクターブとし，半音階は使用しない

- 音符の長さは 4 分音符のみを用いる
- テンポ（メロディを再生する速さ）は一定とする
- メロディ全体の長さは最大 8 小節までとする

### 3.2.6. 作曲画面の仕様

作曲を行うための画面を作曲画面として定義する．作曲画面では音符を自由に配置して作曲を行う．

作曲画面には以下に示すボタンやオブジェクトを配置する．

- 物語遷移ボタン

  物語作成画面へ遷移するボタンである．

  ユーザがこのボタンをクリックすることで，ブラウザ上に表示される画面が物語作成画面に切り換わる．
- 音符

  音色と長さを持ち，動物の画像で表現された音を表すオブジェクトである．
- 譜面

  音符を配置してメロディを形成するスペースである．

  1 オクターブの音階に対応した高さに道があり，その道の上に音符を配置する．

  譜面内には 3 小節分の長さのメロディを表示し，後述する譜めくりと譜戻しで表示する小節の内容を変更出来る．初期状態では 1 小節目から 3 小節目の内容が表示される．

  音符を選択している状態で譜面上にマウスポインタを置くと，音符が配置可能な場所に音符の動物が半透明で表示され，クリックすることでその座標上に音符を配置する．
- 譜めくり

  譜面内に表示されるメロディの内容を 1 小節分進めるボタンである．
- 譜戻し

  譜面内に表示されるメロディの内容を 1 小節分戻すボタンである．
- 楽器ボタン

  作曲に使用する音符を選択するボタン．

  音符に対応した動物のアイコンが描かれており，クリックすることでその音符を選択状態にすることが出来る．
- 消すボタン

  譜面上に配置した音符を削除するボタンである．

  クリックすることで消すボタンが押下状態となり，その状態で譜面上の音符にマウスポインタを置くと，音符上にフォーカスが表示される．クリックすることでそ

の音符を削除する．消すボタンの押下状態はもう一度消すボタンをクリックすることで解除出来る．

- きくボタン

  作成したメロディを最初から最後まで再生するボタンである．

  メロディの再生は，譜面に配置された音符がアニメーションして移動し，譜面内に配置された音階に達した際に対応した高さの音を再生することで行う．

- 呼び出すボタン

  前回までの作業で作成したメロディのデータを読み込むボタンである．

  クリックすると，前回保存したメロディの作品データを読み込み，譜面内に描画する．その際，現在譜面内に配置されている音符は全て削除され，読み込んだ作品データで譜面全体が再描画される．

- 残すボタン

  作成しているメロディのデータを保存するボタンである．

  クリックすると，現在譜面内に形成されているメロディの全ての音符のデータを，作品データとして保存する．

- 作曲保存確認ダイアログ

  メロディを保存した際に表示されるダイアログボックスである．

  作曲保存確認ダイアログにはメロディの保存が正常に完了した旨のメッセージを記述し，ユーザにデータの保存完了を報告する．

- 作曲導入ダイアログ

  作曲を開始する際に表示されるダイアログボックスである．

  作曲導入ダイアログにはメロディを作る手順を示し，ユーザが最初に何をすればいいのかを提示する．

作曲に必要な操作を全てボタンで呼び出すことにより，マウスのみでの操作を実現する．譜面への音符の配置は作曲画面における入力であり，譜面上での音符の表示とメロディの再生は作曲画面における出力である．

作曲画面では，物語作成画面への遷移も可能とする．これは，物語作成画面の仕様で述べたように，物語作成と作曲を並行して行う必要があるためである．物語作成画面へ遷移する際は，作曲画面で作られたメロディのデータはそのままの状態で保持される．

作曲画面で楽器ボタンと譜面，消すボタン，きくボタンを実装することにより，基本的な作曲に関わる要求である U-15 ~ U-18 を満たすことが可能である．そして，呼び出すボタンと残すボタン，作曲保存確認ダイアログを実装することにより，メロディの読み込みと保存に関わる要求である U-19 ~ U-21 を満たすことが可能である．最後に，作曲導入ダイアログを実装することにより，作曲の手順確認に関わる要求である U-22 を満たすことが可能である．

### 3.2.7. 画面遷移

muphic-online の画面遷移図を図 3.1 に示す.

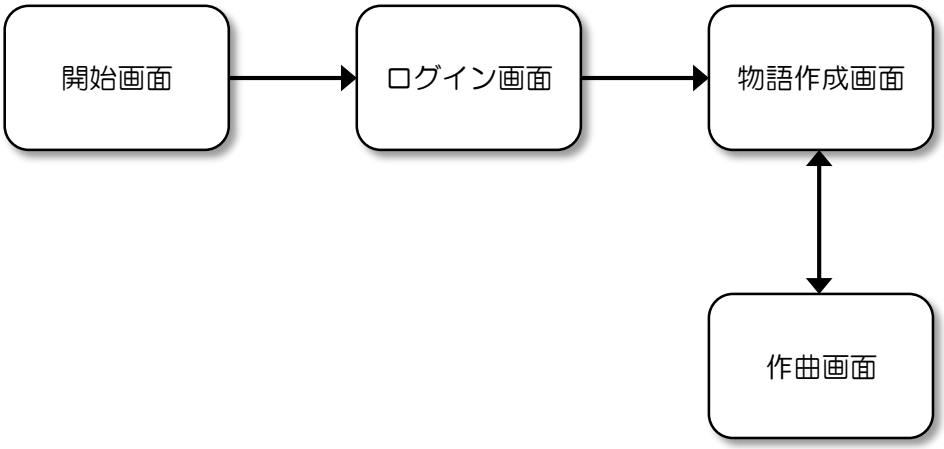


**図 3.1** 画面遷移図

muphic-online を起動すると，最初に開始画面が表示される．開始画面から muphic-online をスタートすると，ログイン画面へ遷移し，ログインを行う．ログインが完了すると物語作成画面へ遷移する．先に作曲画面ではなく物語作成画面へ遷移するのは，作曲と比べて物語作成の方が簡単で児童も遊びやすいと考察したためである．物語作成と作曲は並行して行うことが出来る必要があるため，物語作成画面と作曲画面の間は双方向遷移を可能とする.

### 3.2.8. 各画面のペーパープロトタイプ

3.2.7 で挙げた画面について，物語作成画面と作曲画面のイメージを明確化するためにペーパープロトタイピングを行った．ペーパープロトタイピングとは，紙上にアプリケーションの画面図を描き，実際のアプリケーション使用の流れを紙上でシミュレートすることである．ペーパープロトタイピングを行うことで，画面のイメージや振る舞いを実装前の段階で明確化することが出来る．また，紙に描かれた画面図をペーパープロトタイプと呼ぶ.

物語作成画面のペーパープロトタイプを図 3.2 に示す.

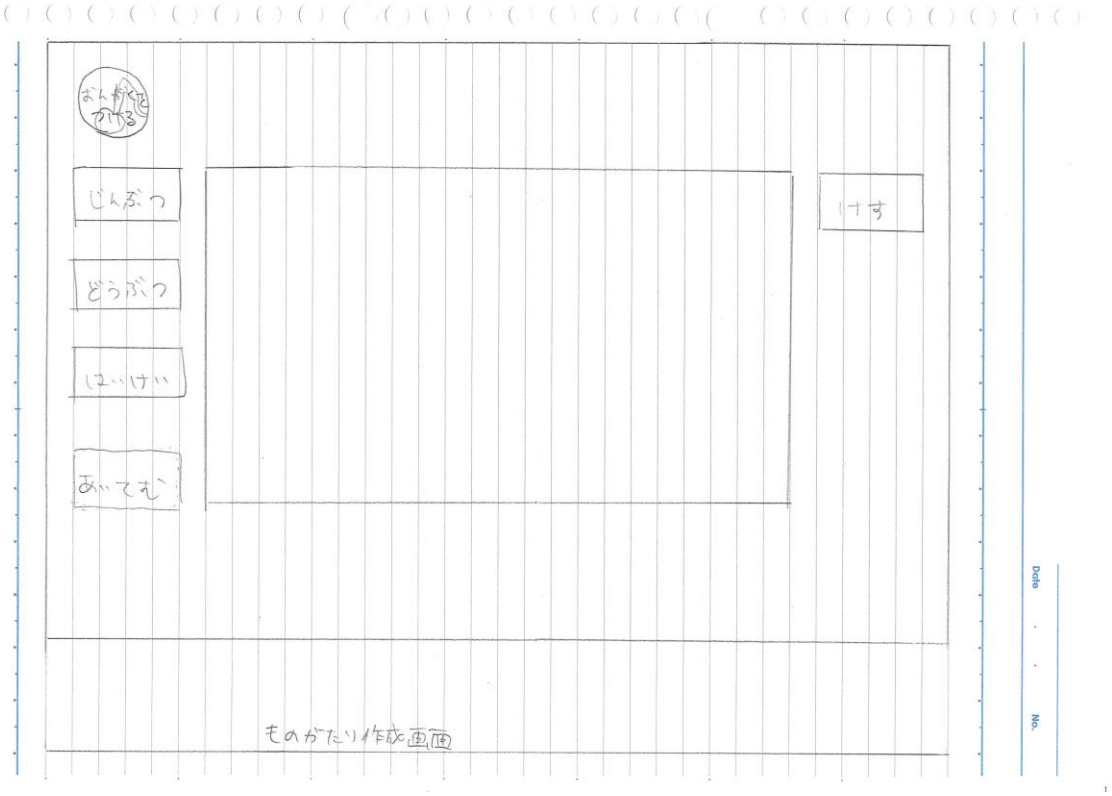


**図 3.2** 物語作成画面のペーパープロトタイプ

物語作成画面では左上に作曲遷移ボタンを配置した．画面中央にはプレビュー画面を配置した. 画面左部には上から順に人物ボタン, 動物ボタン, 背景ボタン, 道具ボタンを配置し, 画面右部には消すボタンを配置した.

物語作成画面に関しては，背景パレットと人物パレットのペーパープロトタイプも作成した．背景パレットのペーパープロトタイプを図 3.3 に示す．

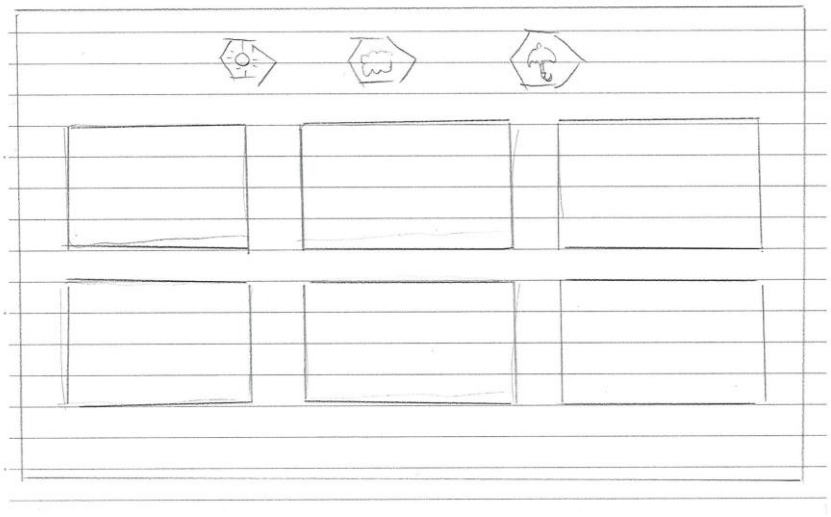

**図 3.3**　背景パレットのペーパープロトタイプ

背景パレットではパレット内に背景イラストを一覧表示する．また，背景パレット上部には天気のボタンを配置した．これらのボタンをクリックすることで，一覧表示されている背景イラストの天気を切り換えることが出来る．

人物パレットのペーパープロトタイプを図 3.4 に示す．

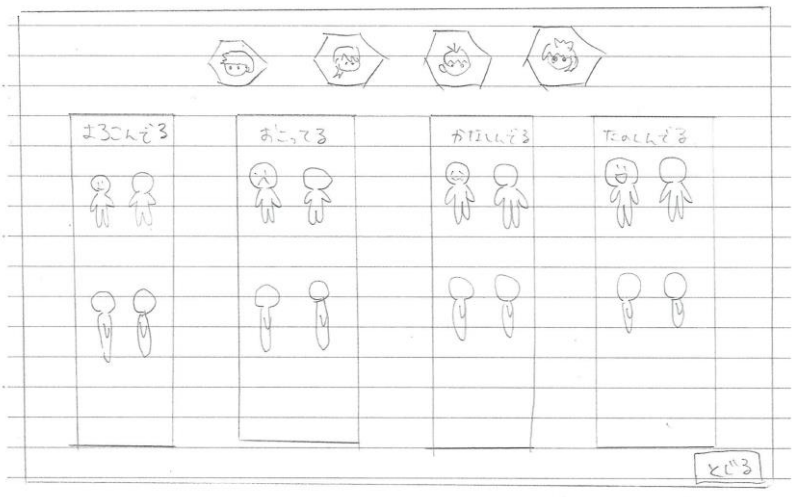


**図 3.4**　人物パレットのペーパープロトタイプ

人物パレットではパレット内に人物イラストを一覧表示する．また，人物パレット上部には人物の種類を表すボタンを配置した．これらのボタンをクリックすることで，一覧表示されている人物イラストの種類を切り換えることが出来る．

作曲画面のペーパープロトタイプを図 3.5 に示す.

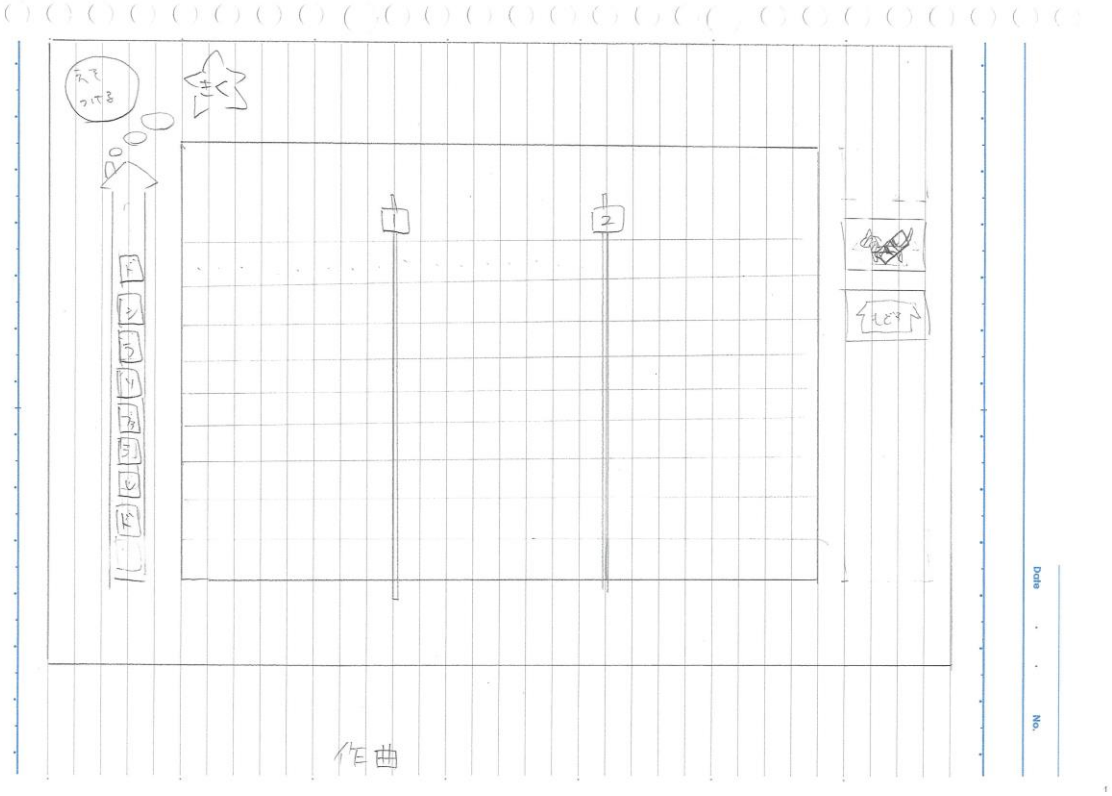


**図 3.5** 作曲画面のペーパープロトタイプ

作曲画面では左上に物語遷移ボタンを配置し，その隣にきくボタンを配置した．画面中央には譜面を配置した．画面右部には楽器ボタンと消すボタンを配置した.

開始画面に関しては，画面の構成が単純なものとなることが予測出来たためペーパープロトタイプの作成は行わなかった．また，ログイン画面に関しては実装の優先度を他の画面よりも低く設定し，設計そのものを開発終盤で行った．よって，ペーパープロトタイプを作成することなく実装に着手した.

### 3.3. サーバ面の仕様

サーバ面はユーザ毎のデータの管理や Web サーバ環境に関わる機能をまとめたものである．ここでは，管理するデータの形式や，オンラインアプリケーションとしての運用に関する仕様をまとめる.

#### 3.3.1. データ管理の仕様

muphic-online では物語作成と作曲の両面について，成果物のデータをサーバ上に保存し，好きな時に再びデータを読み込んで作業を再開出来る機能を実現する．ユーザ毎にそれぞれのデータを保存するためには，ユーザと成果物のデータを紐付ける機能が必要である.

よって，muphic-online ではユーザ名をキーとして，成果物の情報と結びつけるようなデータ構造を用意する．キーとは，データを一意に特定するための情報である．3.2.1 で述べたように，ユーザを一意に特定するためのログインではユーザ名のみを使用するため，入力されたユーザ名をそのままキーとして利用することが出来る．

サーバ側で管理すべきデータを以下にまとめる．

- 物語成果物データ

  物語作成画面で作られた絵の情報を保持したデータである．

  - キー：　ユーザ名
  - 保持する情報：
    - プレビュー画面に配置された背景イラストの情報
    - プレビュー画面に配置された人物イラストの情報
    - プレビュー画面に配置された動物イラストの情報
    - プレビュー画面に配置された道具イラストの情報

- 作曲成果物データ

  作曲画面で作られたメロディの情報を保持したデータである．

  - キー：　ユーザ名
  - 保持する情報：
    - 譜面に配置された音符の情報

### 3.3.2. オンラインアプリケーションとしての仕様

muphic-online を Web アプリケーションとして実現する重要な理由の 1 つに，「あらゆる場所からでもアクセスして使用したいため」という要求がある．このような，どこからでもアクセスして利用できるアプリケーションはオンラインアプリケーションと呼ばれる．オンラインアプリケーションとしての使用要求を満たすため，muphic-online の URL を公開し，ユーザはその URL を Web ブラウザに入力するだけでアクセスが出来ることとする．

### 3.3.3. アーキテクチャの構想

サーバ周辺のアーキテクチャ構想図を図 3.6 に示す.

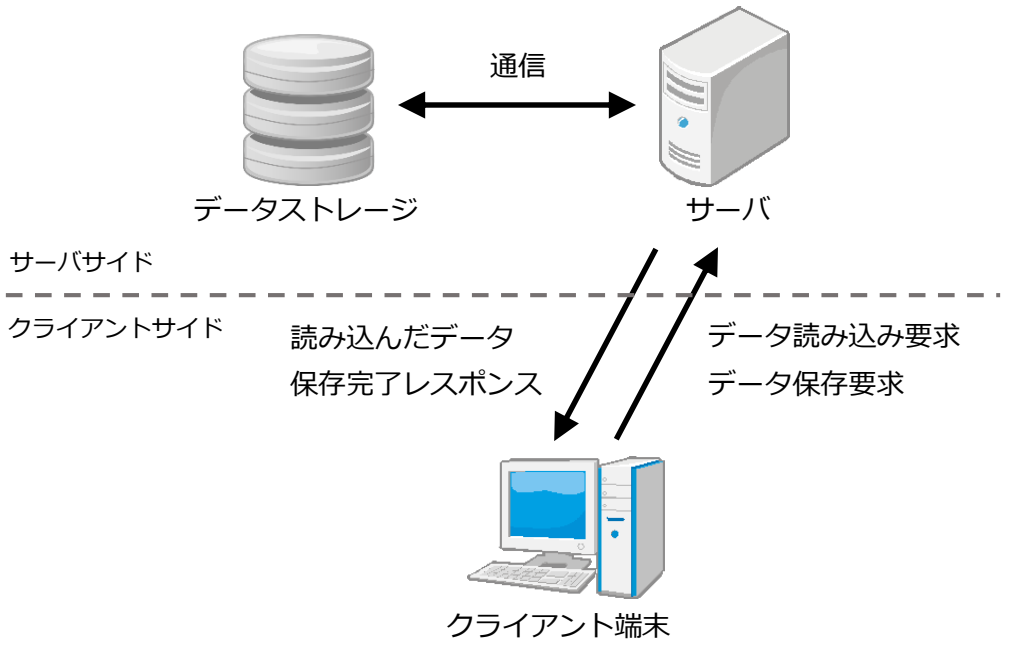


**図 3.6** サーバ周辺のアーキテクチャ構想図

ユーザの成果物のデータはサーバサイドに置かれたデータストレージで管理する．サーバはデータストレージとのデータのやり取り（読み込み・書き込み）を行う機能を持つ．クライアント端末はサーバに対してデータの読み込みや保存のリクエストを送ることが出来る．サーバはクライアント端末からリクエストを受け取ると，データストレージと通信し，データの読み込みや書き込みを行う．そして，データのやり取りの結果をクライアント端末へレスポンスとして返戻する．

# 4. システム仕様を満たすための機能

本章では，仕様を実装するために必要な機能をクライアント面とサーバ面のそれぞれについてまとめる．

## 4.1. クライアント面の実装に必要な機能

クライアント面ではボタンやオブジェクトが配置された画面を定義した．それらの画面を実装するためには画像を描画する機能が必要である．また，作曲画面ではメロディの再生時に音符のアニメーションと音声の再生を行うため，画像のアニメーション機能と，音声ファイルの再生機能が必要となる．画像を用いてボタンを描画する場合，その画像がマウスクリックのイベント機能を有している必要がある．最後に，クライアント面では複数の画面同士での画面遷移が発生するため，画面遷移を簡単に行える機能があると実装が容易になる．

以上の考察から，クライアント面の実装に必要な機能を以下にまとめる．

- 画像の描画機能
- 画像のアニメーション描画機能
- 音声ファイルの再生機能
- マウスクリックのイベント機能
- 画面遷移機能

## 4.2. サーバ面の実装に必要な機能

サーバ面ではユーザの成果物データの管理やオンラインアプリケーションとしての使用方法を定義した．まず，成果物データをクライアント側とサーバ側でやり取りする仕組みを実現するために，クライアント側からサーバ側へのリクエストを処理する機能と，クライアント側とサーバ側の双方向通信機能が必要となる．クライアント側からのリクエストとサーバ側からのレスポンスを通信するために双方向通信機能を用い，クライアント側からのリクエストの内容によってサーバ側で行う処理を選択するためにリクエスト処理機能を用いる．また，成果物データを管理するデータストレージにはデータベース機能を用いる．最後に，オンラインアプリケーションとしてのシステムの実行機能が必要である．

以上の考察から，サーバ面の実装に必要な機能を以下にまとめる．

- クライアント側からサーバ側へのリクエスト処理機能
- クライアント側とサーバ側の間での双方向通信機能
- データベース機能
- オンラインアプリケーション機能

# 5. システムの要素技術

本章では，4 章で示した機能の実装手段として採用した要素技術についてまとめる．

## 5.1. クライアント面の要素技術

クライアント面の実装手段として採用した要素技術についてまとめる．

### 5.1.1. 開発エンジン

クライアント面の要素技術として，enchant.js を採用した[7]．enchant.js は株式会社ユビキタスエンターテイメントがリリースしているオープンソースの JavaScript ゲームエンジンである．JavaScript で記述されたゲームエンジンであるため，enchant.js で記述されたアプリケーションは Web 上で動作する．また，ゲームエンジンということもあり，画像の描画やアニメーション，音声ファイルの再生，マウスイベントの機能をサポートしている．さらに，enchant.js では画面遷移機能もサポートしているため，クライアント面でまとめた機能を全て実装することが可能である．

クライアント面の GUI の細かなレイアウトを調整するために，HTML5 と CSS3 も要素技術として採用した．

### 5.1.2. Web ブラウザ

muphic-online の実行に必要な Web ブラウザについても述べる．muphic-online を実行するための推奨ブラウザには，日本国内で最も普及している Web ブラウザである Internet Explorer を採用した（2013 年 1 月から 2014 年 1 月調べ）[8]．

## 5.2. サーバ面の要素技術

サーバ面の実装手段として採用した要素技術についてまとめる．

### 5.2.1. サーバサイドスクリプト

サーバ側のリクエスト処理を記述する手段として，サーバサイドスクリプトである Node.js を採用した[9]．サーバサイドスクリプトとは，Web サーバ上で動作し，プログラムの実行要求が出されるたびに結果を Web ブラウザに送信するプログラムである．Node.js はサーバサイド JavaScript と呼ばれる．動作のイメージとしては，Web サーバ上では常に Node.js が実行状態となっており，外からリクエストを受信すると，そのリクエストに対応した処理（コンテンツ送信など）を実行する．Node.js ではこの処理の部分を JavaScript で記述出来ることが，サーバサイド JavaScript と呼ばれる所以である．

サーバサイドの処理は PHP 言語で書かれることが多いが，Node.js ならばクライアント側と同じ JavaScript 言語で記述出来るため，PHP 言語を習得する手間がかからず開発効率も落ちることがない．また，クライアント側とサーバ側が両方とも JavaScript で記述さ

れるため，異なるプログラミング言語間でのデータのやり取りを考える必要がない．

### 5.2.2. リアルタイム通信ライブラリ

クライアント側とサーバ側の間での双方向通信機能を実現するため，リアルタイム Web アプリケーション開発ライブラリである Socket.IO を採用した[10]．リアルタイム Web アプリケーションとは，Web ページを切り換えることなくサーバから送られたデータを画面上に反映させることが出来るような Web アプリケーションを示す．Socket.IO は非同期なリアルタイム通信の API を提供する．

Socket.IO を用いてリアルタイム通信を実装することにより，クライアント側とサーバ側の間でのリクエスト・レスポンスの送受信が可能となる．また，Socket.IO はシンプルで理解しやすい API を提供するため，通信部分のコーディングも容易に行える．

### 5.2.3. データベース

データベース機能を実現するため，MongoDB を採用した[11]．MongoDB はデータベースの一種であり，ドキュメント指向型データベースに分類される．ドキュメント指向型データベースではデータベースの中にコレクションというグループを作り，ドキュメントという単位でデータの追加・削除・参照などを行う．

MongoDB ではドキュメントの中に Key-Value 形式でデータを格納するため，JavaScript のハッシュデータや JSON（JavaScript Object Notation，様々なソフトウェアやプログラミング言語間におけるデータの受け渡しに使えるよう設計された軽量データ記述言語の 1 つ）形式のデータと相性が良い．

### 5.2.4. PaaS（Platform as a Service）

オンラインアプリケーション機能を実現するため，Heroku を採用した[12]．Heroku は PaaS（Platform as a Service，ソフトウェアを稼動させるプラットフォームをインターネット経由のサービスとして提供するもの）の一種であり，Web アプリケーションのデプロイ（どこからでもアクセス可能な状態にすること）を可能にする．デプロイした Web アプリケーションには，Heroku から公開される URL を用いることでどこからでもアクセス出来る．

Heroku は Git と呼ばれるソフトウェアのバージョン管理システムを利用することで容易にデプロイを行える．また，アドオンを追加することで前述した MongoDB への対応も可能となる．

# 6. システムの開発環境と実行環境

本章では，クライアント面とサーバ面のそれぞれについて，開発環境と実行環境をまとめる．

## 6.1. クライアント面

- 開発環境
  - enchant.js（JavaScript）　：ユーザインタフェース開発エンジン
  - HTML5　：ユーザインタフェース開発言語
  - CSS3　：ユーザインタフェース開発言語
- 実行環境
  - Internet Explorer（バージョン 9）　：Web ブラウザ

## 6.2. サーバ面

- 開発環境
  - Node.js（JavaScript）　：サーバサイド開発言語
  - Socket.IO（JavaScript）　：サーバサイド開発言語
  - MongoDB　：データベース
- 実行環境
  - Heroku　：PaaS（Platform as a Service）

# 7. システムの実装

本章では，要素技術を用いたシステムの実装についてまとめる．

## 7.1. クライアント面の実装

クライアント面の実装では，開始画面，ログイン画面，物語作成画面，作曲画面の実装を行った．各画面の実装と，実装方法についてまとめる．

### 7.1.1. 開始画面

開始画面の実装結果を図 7.1 に示す．


**図 7.1** 開始画面

開始画面には muphic-online をスタートするための「はじめるボタン」のみを配置した．「はじめるボタン」をクリックすることでログイン画面へ遷移する．

### 7.1.2. ログイン画面

ログイン画面の実装結果を図 7.2 に示す.


**図 7.2** ログイン画面

ログイン画面には「文字ボタン」「消すボタン」「テキストボックス」「名前決定ボタン」を配置した.

文字ボタンはアルファベットの小文字と大文字のものを実装した. 文字ボタンをクリックすると, それぞれのボタンに書かれたラベルの文字が入力され, テキストボックスに表示される. 消すボタンをクリックすると, 入力した文字を 1 文字削除する.

テキストボックスは, ユーザからの入力が 1 文字もない場合は「したのボタンをおしてね」というメッセージを表示し, ユーザに文字入力を促す配慮をしている. テキストボックス内には仕様で定めたように最大 15 文字まで文字を表示する.

名前決定ボタンは, ユーザからの入力が 1 文字もない場合は半透明表示し, クリックを受け付けない. ユーザからの入力が 1 文字以上ある場合にクリックを受け付ける. 名前を入力して名前決定ボタンをクリックすることで物語作成画面へ遷移する.

### 7.1.3. 物語作成画面

物語作成画面の実装結果を図 7.3 に示す.


**図 7.3** 物語作成画面

物語作成画面には「作曲遷移ボタン」「背景ボタン」「人物ボタン」「動物ボタン」「道具ボタン」「消すボタン」「プレビュー画面」「呼び出すボタン」「残すボタン」を配置した.

作曲遷移ボタンをクリックすることで作曲画面へ遷移する.

背景ボタン，人物ボタン，動物ボタン，道具ボタンをクリックすると，それぞれのパレットを呼び出す．パレットの例として，人物パレットを図 7.4 に示す.

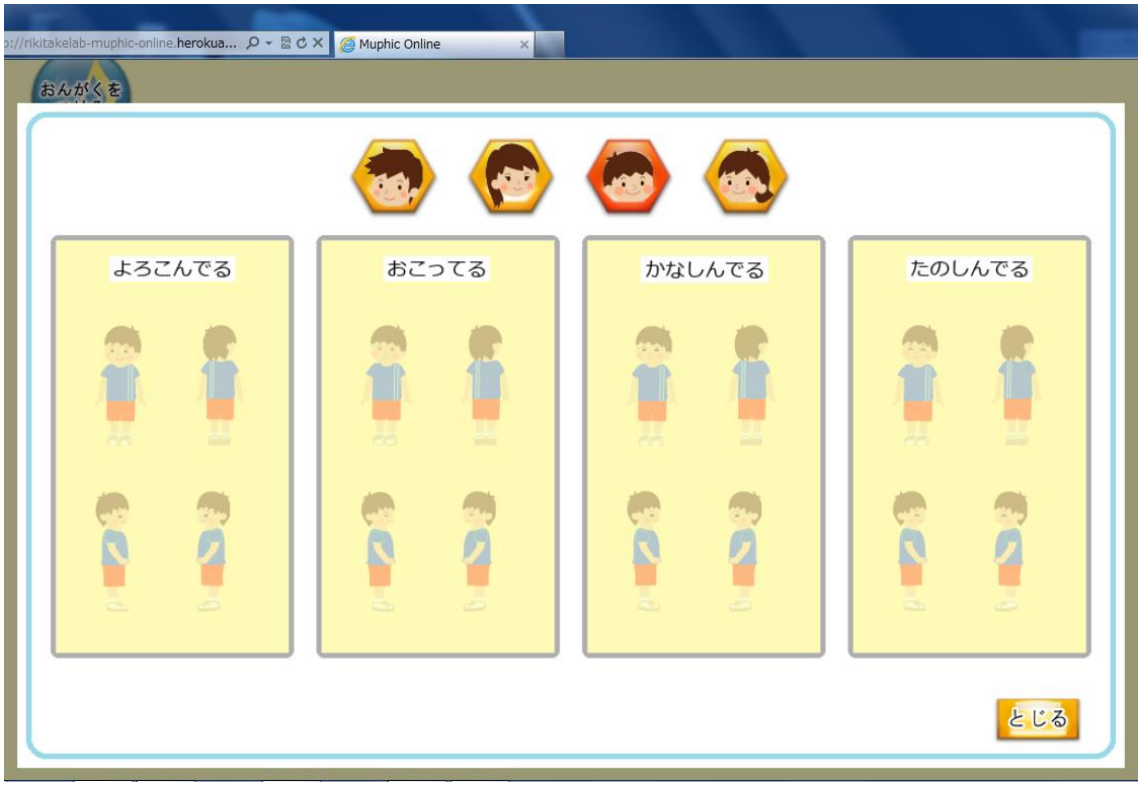


**図 7.4**　人物パレット

人物イラストには「お兄さん」「お姉さん」「男の子」「女の子」の画像を用意した．それぞれについて「喜」「怒」「哀」「楽」の 4 種類の表情と「前」「後」「左」「右」の 4 種類の方向を実装した．人物パレットの上部には人物イラストの種類のアイコンが描かれたボタンを配置した．ボタンをクリックすると，それぞれの種類のイラストが一覧表示される．その中から好きなイラストをクリックすることにより，イラストを選択することが出来る．パレットには閉じるボタンも配置し，クリックすることでパレットを閉じることが出来る．

イラストを選択した状態でプレビュー画面上にマウスポインタを置くと，その場所にイラストが半透明表示される．クリックすると，その座標上にイラストを配置する．

消すボタンをクリックすると消すボタンは押下状態となり，消すボタンの色が赤色に変化する．その状態でプレビュー画面上の人物イラスト，動物イラスト，道具イラストをクリックすると，イラストを削除出来る．

呼び出すボタンをクリックすると，前回保存した物語のデータを読み込んでプレビュー画面に表示する．残すボタンをクリックすると，現在のプレビュー画面の内容を保存する．

### 7.1.4. 作曲画面

作曲画面の実装結果を図 7.5 に示す.

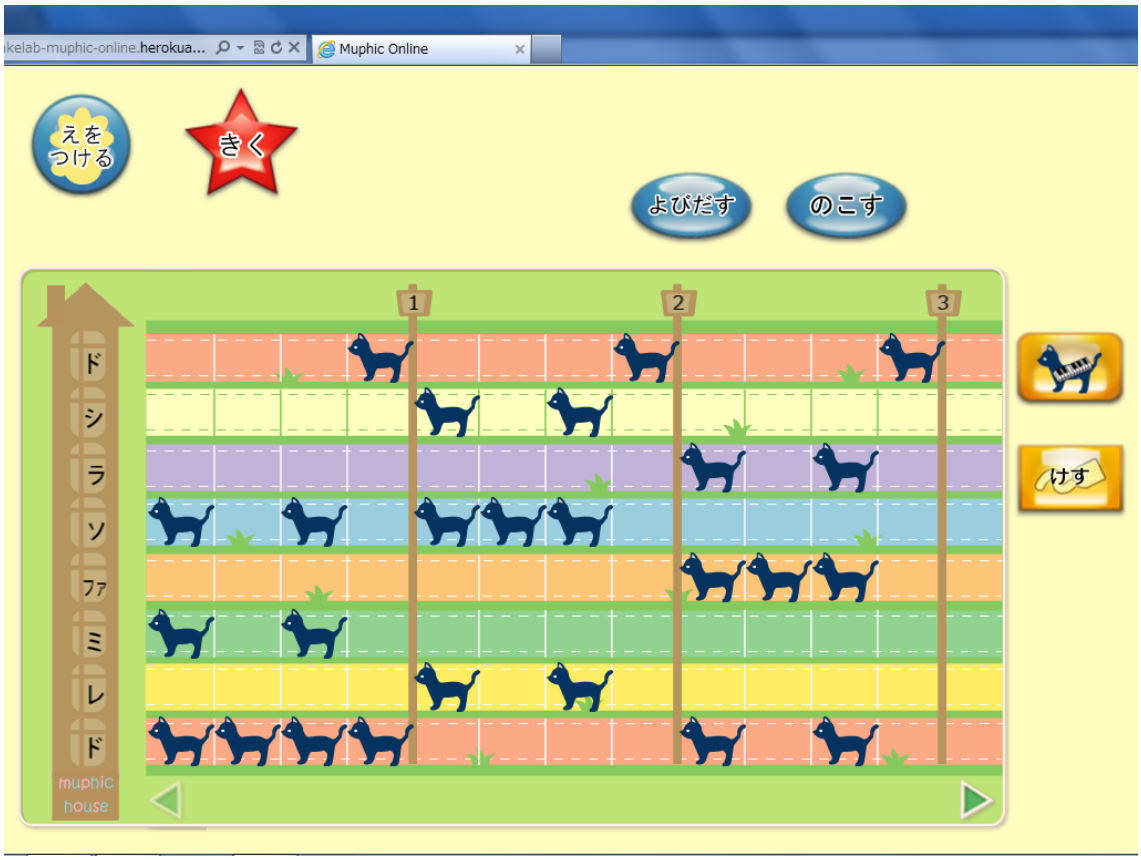


**図 7.5** 作曲画面

作曲画面には「物語遷移ボタン」「譜面」「譜めくり」「譜戻し」「楽器ボタン」「消すボタン」「きくボタン」「呼び出すボタン」「残すボタン」を配置した.

物語遷移ボタンをクリックすることで物語作成画面へ遷移する.

譜面は画面中央に配置されたスペース全体である. 譜面左側には音階を表す家が描かれており, 音の高さに対応するように譜面には道が描かれている. この道の上に音符を配置してメロディを形成する. 木のボードに描かれた数字は小節の番号を示す. 譜面右下の三角形のボタンが譜めくりであり, 左下の三角形のボタンが譜戻しである.

楽器ボタンはネコのアイコンが描かれたボタンである. クリックすることで押下状態となり, 楽器ボタンが赤色に変化する. その状態で譜面上にマウスポインタを置くとその場所に音符が半透明表示され, クリックするとその座標上に音符を配置する.

消すボタンは楽器ボタンの下に配置されている. クリックすることで押下状態となり, 消すボタンが赤色に変化する. その状態で譜面上の音符をクリックすると, 音符を削除出来る.

きくボタンはクリックすることでメロディを最初から最後まで再生出来る. メロディ再生中は全ての音符が左方向へアニメーション移動し, 家の窓に達した際に音声を再生する.

メロディ再生中は全てのボタンのクリックを不可とし，譜面の操作も不可とする．尚，譜面上に音符が 1 つもない場合はきくボタンのクリックを不可とする．

呼び出すボタンをクリックすると，前回保存したメロディのデータを読み込んで譜面に表示する．残すボタンをクリックすると，現在の譜面の内容を保存する．

実装した作曲導入ダイアログを図 7.6 に示す．


**図 7.6**　作曲導入ダイアログ

作曲導入ダイアログは初めて作曲画面に遷移した際に表示される．ダイアログ内ではメロディを作るための初歩の手順として，音符の配置方法を示している．テキストと画像を組み合わせた説明により，具体的な操作を提示している．ダイアログ下部の「はいボタン」をクリックすると，ダイアログを閉じて作曲を開始可能とする．

### 7.1.5. 実装方法

クライアント面の実装に関して，実際の実装方法についてまとめる．

- 疑似的なマウスオーバー機能の実装

  Web ページなどで，リンクが設定された画像や文字列にマウスカーソルを重ねると，リンク先の URL が表示されるといった機能がある．このような機能をマウスオーバーと呼ぶ．

  muphic-online ではマウスオーバーを用いて，ボタンの上にマウスカーソルを置くとボタンの色を変化させるといった機能を実装したいと考えた．ボタンにマウスオーバーを実装することで，単なる画像ではないという印象をユーザに与えることが出来ると考察した．しかし，enchant.js ではマウスオーバーイベントをサポートしていない．そこで，HTML5 の機能を使用して疑似的なマウスオーバー機能を実現した．

  enchant.js で作成したアプリケーションは，html ファイルの body タグの中で実行され，動作する．html ファイルの body タグには onmousemove 属性が含まれている．onmousemove 属性では，マウスなどのポインティングデバイスのカーソルが移動した際に起動するスクリプト（関数）を指定出来る．スクリプトへ渡すイベント引数からはカーソルの x 座標値と y 座標値を得ることが出来る．よって，muphic-online を実行する html ファイルの body タグに，onmousemove 属性と対応スクリプトを記述し，スクリプトの実行を通してマウスカーソルの座標値をリアルタイムに取得することを可能とした．html ソースコードの body 部の抜粋を図 7.7 に示す．

```
<body id="app" onmousemove="getScreenPoints()">
</body>
```

**図 7.7** html ソースコードの body 部

onmousemove イベントに対し，実行するスクリプトとして getScreenPoints を記述している．getScreenPoints 関数のソースコードを図 7.8 に示す．

```
var clientX = null;
var clientY = null;

var getScreenPoints = function() {
    clientX = event.clientX;
    clientY = event.clientY;
}
```

**図 7.8** getScreenPoints 関数ソースコード

最初にマウスカーソルの座標値を格納するグローバル変数 clientX と clientY を定義してある．getScreenPoints 関数が実行されると，イベント引数から x 座標値と y 座標値をそれぞれ取得し，グローバル変数へ格納する．これにより，muphic-online のソースコード内ではグローバル変数 clientX と clientY を参照することでいつでもマウスカーソルの座標値を得ることが出来る．

マウスオーバーを実装したいボタンやスプライト（画面に描画される画像）では常にマウスカーソルの座標値を監視し，自らが描画されている座標の範囲内にマウスカーソルの座標が入った際に画像のデザインを変化させるといった処理を実装することで，ボタンなどへのマウスオーバー機能を実現した．

マウスオーバー機能は muphic-online の各画面に配置された全てのボタンや，物語作成画面におけるプレビュー画面上でのイラストの配置，作曲画面における譜面上での音符の配置に応用している．

- 物語作成画面でのイラストの配置

  物語作成画面ではプレビュー画面とマウスオーバー機能，マウスクリックイベントを組み合わせてイラストの配置機能を実装した．

  プレビュー画面にはマウスオーバー機能を持たせ，マウスカーソルの座標を常に監視させる．マウスカーソルがプレビュー画面内に入った際に，選択中のイラストをスプライトとしてプレビュー画面の上に描画する．スプライトは半透明で表示し，x 座標値と y 座標値には常にマウスカーソルの x 座標値と y 座標値をそれぞれ与えることにより，半透明のイラストの画像がマウスカーソルに追従する動きを実現した．スプライトはマウスクリックイベントの機能も持つ．ユーザがプレビュー画面をクリックする際は，実際には半透明のスプライトをクリックすることになる．よって，スプライトのマウスクリックイベントでプレビュー画面へイラストの配置を通知し，プレビュー画面はクリックされた座標上に選択中のイラストのスプライトを描画する（図 7.9 参照）．

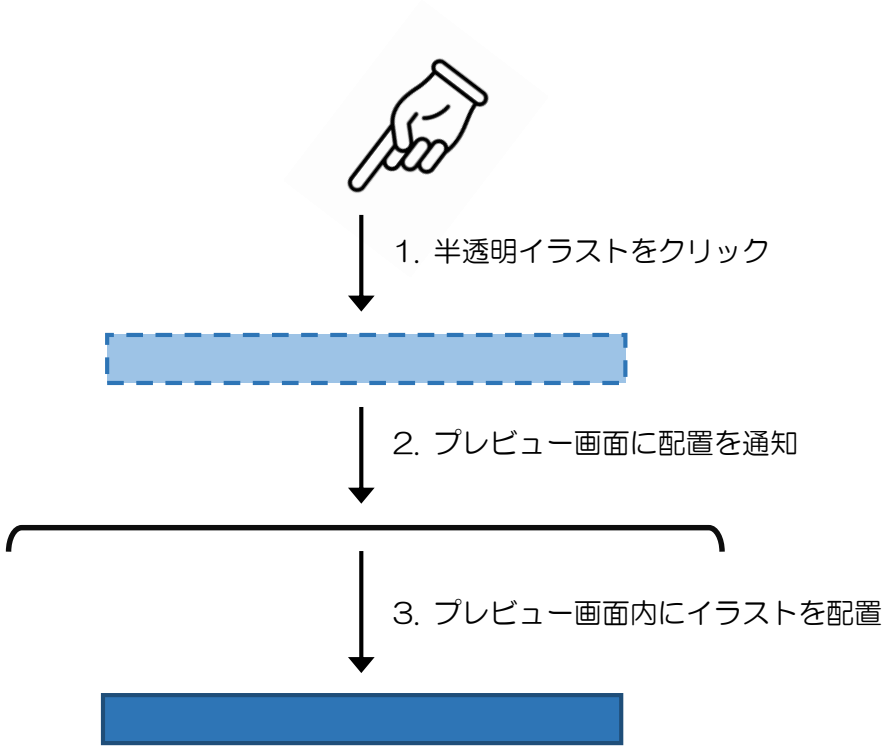


**図 7.9** プレビュー画面へのイラスト配置のイメージ

半透明イラストをクリックした際に実行される_createIllust 関数のソースコードを図 7.10 に示す.

```
_createIllust : function() {
    var x = this._getX();
    var y = this._getY();
    this._getParent()._createIllust(x, y);
},
```

**図 7.10** 半透明イラストの_createIllust 関数

半透明イラストの_createIllust 関数では，クリックされた時点での半透明イラストの x 座標値と y 座標値（マウスカーソルの x 座標値と y 座標値）を取得する．そして，その座標値をプレビュー画面の_createIllust 関数に引数として渡し，実行している．プレビュー画面の_createIllust 関数実行が，半透明イラストからプレビュー画面へのイラスト配置通知を意味している．プレビュー画面の_createIllust 関数のソースコードを図 7.11 に示す.

```
_createIllust : function(x, y) {
    this._illustFocus._resetPoint();
    this._illustFocus._removeFromScene();
    this._getController()._createIllust(x, y);
    this._getParent()._setDisableState();
},
```

**図 7.11**　プレビュー画面の_createIllust 関数

プレビュー画面の_createIllust 関数では，半透明イラストを画面上から削除してから，プレビュー画面上に選択中のイラストのスプライトを配置する．このような関数の実行手順を通して，プレビュー画面内にイラストを配置する．

- 作曲画面での音符のグルーピング

作曲画面ではメロディを再生する際に譜面上の音符全てをアニメーション移動させる．この時，音符 1 つ 1 つをアニメーション移動させた場合，大量のスプライトを一斉にアニメーション描画することによって画面全体への負荷が大きくなるのではないかと考察した．よって，全ての音符を 1 つの大きなスプライトとしてグルーピングした．これにより，スプライト 1 つのみをアニメーション描画するだけで，あたかも音符全体がアニメーション移動しているかのように見せることが出来る（図 7.12 参照）．

また，譜めくりや譜戻しで譜面内に表示する小節の内容を切り換える際も，音符 1 つ 1 つの x 座標を変更せず，スプライト 1 つの x 座標を変更するだけで音符の表示を切り換えられる．

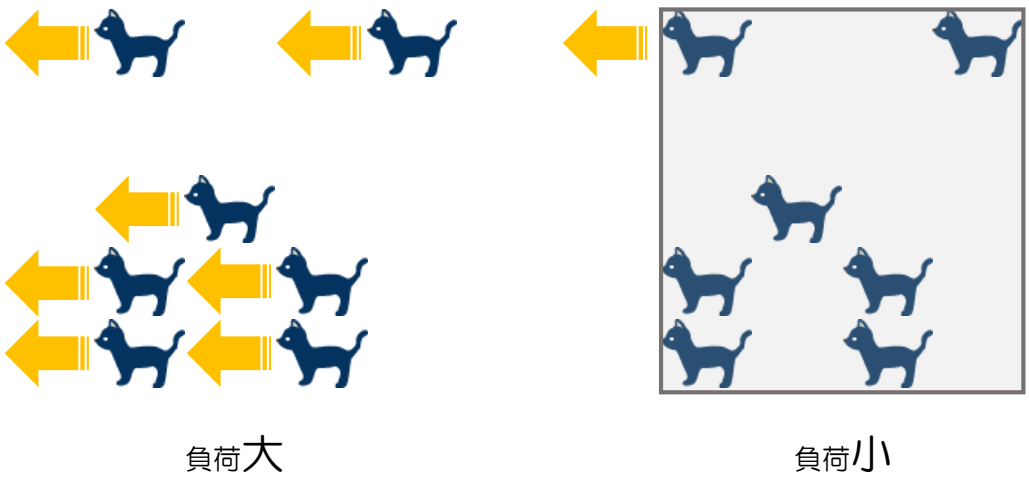


**図 7.12**　スプライトのグルーピングのイメージ

## 7.2. サーバ面の実装

サーバ面の実装では，システム全体のアーキテクチャ，データベース，データベースアクセス API の実装を行った．それぞれの実装についてまとめる．

### 7.2.1. システムアーキテクチャの実装

サーバ周辺のアーキテクチャの構想を基に，システム全体のアーキテクチャを実装した．実装したシステムアーキテクチャを図 7.13 に示す．

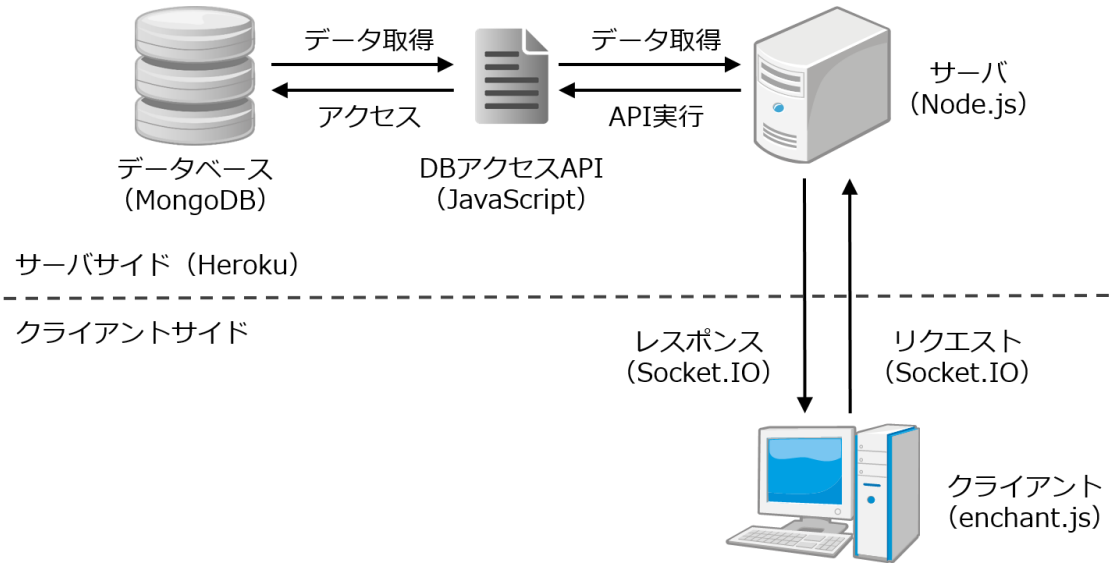


**図 7.13** システムアーキテクチャ

クライアント側とサーバ側の通信プロセスについて順序立てて述べる．最初に，クライアント側はデータの保存や読み込みを行う際にサーバへリクエストを送信する．リクエスト送信メソッドの作成に Socket.IO を用いた．サーバ側は受信したリクエストの内容に応じて，実行するデータベースアクセス API を選択する．リクエストと実行する API の対応の記述に Node.js を用いた．API とは具体的に言えば JavaScript で記述されたメソッドであり，データベースへのデータの保存・参照などを行う処理を記述した．実行された API はデータベースを操作し，操作した結果を保持する．保存操作であれば，データベースに正しくデータを保存出来たか否かが操作結果であり，参照操作であれば，データベースを参照して得られたデータが操作結果である．操作を完了した API はサーバへ操作結果を返戻する．そして最後に，サーバは API から返戻された結果をクライアント側へレスポンスとして送信する．リクエストと同様に，レスポンス送信メソッドの作成にも Socket.IO を用いた．

サーバ面全体は Heroku 上で動作する．クライアント側は Heroku から公開された URL を Web ブラウザに入力することで，どこからでも muphic-online にアクセスして遊ぶことが出来る．本研究の実装では Heroku を使用したが，Node.js，Socket.IO，MongoDB を扱

える実行環境であれば，Heroku 以外の PaaS やサーバ環境でもこのシステムアーキテクチャで運用が可能である．

### 7.2.2. データベースの実装

ユーザの成果物のデータを保存するためのデータベースを実装した．「物語成果物データ」と「作曲成果物データ」の 2 種類について，スキーマ（関係型データベースにおけるテーブル）を定義し，MongoDB を用いて実装した．以下にその定義についてまとめる．

- 物語成果物データ

  ユーザ名をキーとして，物語のデータを一意に特定可能なスキーマを構築した．

  物語成果物データは「物語の背景イラストの情報」と「物語に使用されている人物イラスト, 動物イラスト, 道具イラストの情報」に分割してスキーマを構築した．

  背景イラストのデータスキーマを表 7.1 に示す．

**表 7.1** 物語の背景イラストのデータスキーマ

| ユーザ名 | 風景情報 | 天気情報 |
|---|---|---|

物語の背景は，風景と天気の画像を重ねて表示している．よって，風景画像を特定するための風景情報と，天気画像を特定するための天気情報をそれぞれ保存するよう定義した．風景情報には，風景の種類である「森」「水辺」「町」「室内」を表す文字列が格納される．天気情報には，天気の種類である「晴れ」「曇り」「月夜」を表す文字列が格納される．

人物，動物，道具イラストのデータスキーマを表 7.2 に示す．

**表 7.2** 物語の人物，動物，道具イラストのデータスキーマ

| ユーザ名 | イラスト種類情報 | 特定情報 1 | 特定情報 2 | 特定情報 3 | x 座標 | y 座標 |
|---|---|---|---|---|---|---|

イラスト種類情報には，イラストの種類である「人物」「動物」「道具」を表す文字列が格納される．イラストを特定するために必要な情報として特定情報を 3 種類まで定義した．特定情報には，イラストを特定するために必要な情報が格納される．例えば，人物イラストの場合は「人物の種類」「表情」「方向」の 3 つの情報がそれぞれ特定情報として格納される．動物イラストの場合も人物イラストと同様である．道具イラストの場合は「道具の種類」「道具」の 2 つの情報が特定情報として格納されるため，特定情報 3 が空欄で構わない．そして，プレビュー画面上に置かれた際の x 座標値と y 座標値を定義した．これらの情報を利用することで，人物，動物，道具イラストを特定することが可能である．

- 作曲成果物データ

  ユーザ名をキーとして，メロディのデータを一意に特定可能なスキーマを構築した．メロディは複数の音符から構成されるため，音符を表すスキーマを構築した．音符のデータを複数保存することにより，1 つのメロディのデータを保存することが出来る．

  音符のデータスキーマを表 7.3 に示す．

**表 7.3** 作曲の音符のデータスキーマ

| ユーザ名 | 動物 | 小節 | 拍 | 音階 |
|---|---|---|---|---|

音符を構成するためには「音色」「小節」「拍」「音の高さ」の情報が必要である．音符の音色は動物で表現されているため，動物の名前情報をスキーマに含めた．

### 7.2.3. データベースアクセス API の実装

データベースの操作を行う API（JavaScript メソッド）の実装を行った．データベースの操作に関しては，「保存」「参照」「削除」の 3 種類が重要であると考えた．よって，これら 3 種類の操作を行うメソッドを実装した．MongoDB では Key-Value 形式でデータを管理するため，JavaScript でハッシュデータを扱うのと同じ感覚でデータベースも操作出来る．よって，API メソッドの実装も容易に行うことが出来た．取得したデータはキーを指定するだけで値を取り出すことが可能である．

クライアントからのリクエストと各 API の対応を Node.js を用いて記述した．対応にはメッセージと呼ばれる任意の文字列を使用した．例えば，「save」というメッセージを受信した場合は保存 API メソッドを実行するよう定義した場合は，クライアント側は「save」という文字列を添えてリクエスト送信を行うだけでデータベースへのデータの保存を実行出来る．「保存」「参照」「削除」の処理を行う各 API メソッドを，データベース上の全てのコレクションに対して実装した．

# 8. システムの考察

本章では，実装したシステムのクライアント面とサーバ面の考察についてまとめ，全体としてユーザの要求を満たすシステムが開発出来たかを考察する．

## 8.1. クライアント面の考察

クライアント面の仕様の充足と，実装の工夫点についての考察をまとめる．

### 8.1.1. クライアント面における仕様の充足について

実装したシステムがクライアント面の仕様を満たせているか考察する．

インタフェースに関しては，実装した全ての画面において操作をマウスのみで完結出来るよう実装したため，「マウスのみの簡単な操作」という仕様は満たすことが出来た．システム内のテキストやボタンのラベルは基本的に平仮名で記述したため，「平仮名によるテキスト」という仕様も満たすことが出来た．

ログインに関しては課題が残った．ログイン画面で平仮名と片仮名の文字ボタンを実装することが出来ず，結果としてユーザ名にはアルファベットのみが使用可能となった．平仮名と片仮名に対してアルファベットには不慣れな児童が多いと考えられるため，アルファベットのみしかユーザ名に使用出来ないログイン機能は児童にとって低価値である．最低限のログインは可能であるが，改良の余地がある．

物語作成に関してはイラストの選択・配置・削除の機能を実装し，物語の作成が行える段階まで実装出来た．イラストに関しても，muphic で使われていたリソースを再利用することで豊富な種類の画像データを用意することが出来，様々な組み合わせが可能な物語作成の機能をユーザに提供出来ると考えられる．読み出しと保存の機能に関しても，試用した結果正しく実装されていることが確認出来た．しかし，物語保存確認ダイアログの実装は出来ていないため，ユーザは物語の保存が正しく完了されたか確認することが出来ない．また，物語導入ダイアログも実装出来ていないため，ユーザは物語の作り方を自分で覚えながらシステムを使用しなければならない．よって，物語作成機能としては使えるシステムが完成しているものの，重要度の高くない小さな機能が実装されていないという状態である．物語作成としての仕様は概ね満たせているが，未実装の機能を実装することで物語作成機能の価値をより高めることが出来ると考えられる．

作曲に関しては音符の選択・配置・削除の機能を実装し，メロディの作成が行える段階まで実装出来た．読み出しと保存の機能に関しても，試用した結果正しく実装されていることが確認出来た．作曲画面では作曲導入ダイアログを実装することが出来，物語作成と比べてやや難しい作曲の手順を確認出来ることはユーザにとって価値があると考えられる．しかし，物語作成画面と同様に作曲保存確認ダイアログの実装は出来ていない．よって，保存の完了が確認出来ないという欠点があるものの，作曲の仕様はほぼ満たすことが出来たと考えられる．

各機能の仕様の達成度から，物語作成と作曲を行えるシステムが開発出来たと考えられる．しかし，muphic-online の重要な機能である読み出しと保存に関わるログイン機能の完成度が低いのは問題である．本格的な運用を考える場合は，ログイン機能の改良を優先して行う必要がある．

### 8.1.2. クライアント面における実装の工夫点について

クライアント面における実装の工夫点について考察する．

- オブジェクト指向プログラミング

  クライアント面の実装に使用したゲーム開発エンジン enchant.js はオブジェクト指向プログラミングをサポートしている．したがって，クラスや継承といったオブジェクト指向の概念を利用出来る．

  muphic-online ではボタンや画像を全てオブジェクトとして実装しているため，大量のオブジェクトが存在する．よって，オブジェクトの共通機能を抽出してクラス化し，継承を駆使して実装を行うことで効率の良い開発を追求した．例えば，muphic-online の全てのボタンにはマウスオーバーでボタンの色を変化させるという機能がある．このマウスオーバー機能は全てのボタンで共通な機能であるため，抽出してクラス化する．そして，各ボタンを実装する際はボタンのスーパークラスを継承してサブクラスとして実装する．これにより，新たなボタンを作る度にマウスオーバーの処理を記述する必要がなくなり，新しいボタン特有の処理のみを記述すれば済むようになる．このような実装方法は，1 つあたりのクラスのコード記述量を少なく出来る効果もある．他と異なる部分のみを記述する差分プログラミングを行うことで，開発の効率を向上させる効果がある．

- MVC モデルを採用した実装

  ソフトウェア設計モデルの 1 つに MVC モデルが存在する．これは，データや値を管理する「Model」，値の出力を担当する「View」，入力を受取って Model や View を制御する「Controller」の 3 要素の組み合わせでシステムを実装する方式である．

  muphic-online のプログラミングではこの MVC モデルを採用した．MVC モデル化により，データを操作するロジック部分とデータを出力する GUI 部分との明確な切り離しが可能となる．これにより，ロジック部分のユニットテストのテストコード記述が容易になるという利点がある．ロジック処理と GUI 処理が密結合したソースコードのユニットテストコードを記述することは困難である．しかし，ロジック処理と GUI 処理を疎結合させることが出来れば，ロジック部分のみのユニットテストコードを記述することが容易になる．これは，ユニットテストの実施を促す効果があると同時に，明確な機能の切り離しによってソースコードの見通し

を良く出来る効果もある.

muphic-online における MVC モデルを用いた実装の一例として, 作曲画面の音符の実装を図 8.1 に示す.

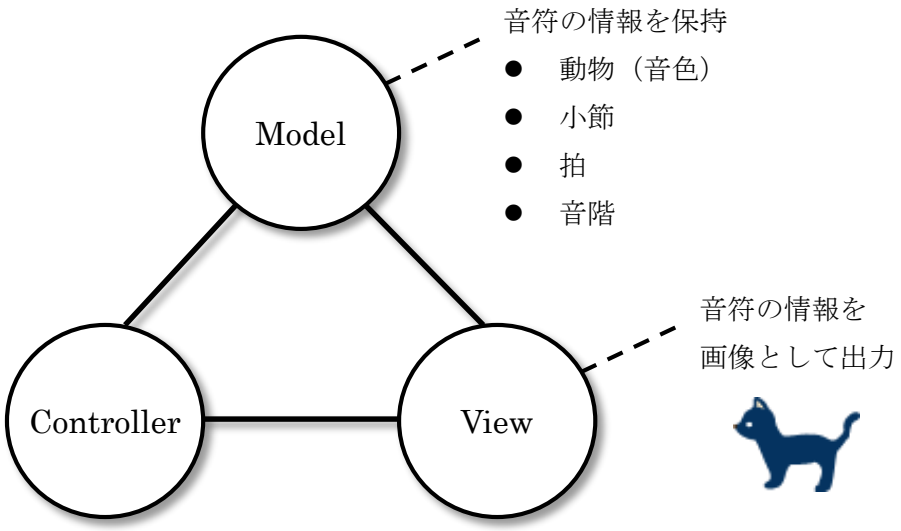


**図 8.1** MVC モデルによる作曲画面の音符の実装

Model では, 音符を構成する情報である「動物（音色)」「小節」「拍」「音階」をデータとして保持する. View は, Model の情報を基にして譜面上に動物を表示し, 音符を画像として出力する. View の x 座標は Model の「小節」と「拍」, y 座標は Model の「音階」の情報を用いて計算する. 音符はボタンなどと違ってユーザからの入力を受け取るオブジェクトではないため, Controller の責務はない.

- データベースとの連携を考慮した実装

  データベースの実装に使用した MongoDB は, 連想配列形式のデータを管理する. この特徴を活かし, クライアント面では物語やメロディの作成に必要なデータは全て連想配列形式で実装を行った. 例えば, 音符データは「動物 (音色)」「小節」「拍」「音階」をキーとする連想配列形式で実現した. このような実装を行うことにより, 物語やメロディのデータを直接データベースへ保存し, またデータベースから読み込んだ物語やメロディのデータを直接使用することも可能となる. MongoDB の特徴を活かした合理的な実装を行い, データベースの処理部分の簡易化を図った.

## 8.2. サーバ面の考察

サーバ面の仕様の充足と，実装の工夫点についての考察をまとめる．

### 8.2.1. サーバ面における仕様の充足について

実装したシステムがサーバ面の仕様を満たせているか考察する．

クライアント面の物語作成と作曲について読み出しと保存が正しく実行されることを確認出来たため，システムアーキテクチャやデータベースの実装は正しく行うことが出来たと考えられる．しかし，データベースの実装に関しては合理化の余地がある．データベースのスキーマの構成がクライアント面のソースコードの記述に依存している箇所があるため，まずはクライアント面のソースコードの再構成・合理化を行う必要がある．

Heroku のデプロイ機能により，オンラインアプリケーションとしての運用も可能であることが確認出来た．

各機能の仕様の達成度から，サーバ面の仕様は満たすことが出来たと考えられる．

### 8.2.2. サーバ面における実装の工夫点について

サーバ面における実装の工夫点について考察する．

- Node.js と MongoDB の組み合わせ

  Node.js を使用したことにより，サーバの振る舞いを JavaScript 言語で記述することが出来たが，データベースを操作するための API メソッドも JavaScript 言語で記述することが出来た．これにより，データベースの操作に関する実装を迅速且つ容易に行うことが出来，サーバ面の実装を効率良く行うことが出来た．

## 8.3. システム全体における要求の充足について

実装したシステムがユーザの要求を満たせているか考察する．「アカウント」「物語」「作曲」の 3 種類の分類とインタフェースについて，要求の充足に関する考察を述べる．

「アカウント」では，名前を入力してログインを行いたいという要求があった．この要求に関しては，ログイン画面での名前入力で平仮名，片仮名文字を使用出来ないため，要求を満たすことは出来なかったと考えられる．

「物語」では，全体としてイラストを配置して物語を作りたいという要求があった．この要求に関しては，物語作成画面で物語を作るための最低限の機能を実装出来たため，要求を満たすことが出来たと考えられる．しかし，物語成果物の保存確認機能や物語作成手順の紹介機能は実装出来なかった．よって，「物語」に関する全ての要求を満たすことは出来なかったと考えられる．

「作曲」では，全体として音符を配置してメロディを作りたいという要求があった．この要求に関しては，作曲画面でメロディを作るための最低限の機能を実装出来たため，要求を

満たすことが出来たと考えられる．しかし，作曲成果物の保存確認機能は実装出来なかった．よって，「作曲」に関する全ての要求を満たすことは出来なかったと考えられる．

インタフェースに関しては，クライアント面の全ての画面についてあらゆる操作をマウスのみで完結出来るよう実装を行った．これにより，簡単に操作をしたいという要求を満たすことが出来たと考えられる．また，画面の実装にはかわいらしいイラスト画像を使用し，画面内のテキストにも基本的に平仮名文字のみを用いた．よって，児童にとって親しみやすいユーザインタフェースを実現出来たと考えられる．

以上の考察から，システム全体としてはユーザの要求を概ね満たすことが出来たと考えられる．

# 9. 今後の展望

本章では，開発したシステムに関して今後実践すべきことや有用性の検証内容についてまとめる．

## 9.1. システムの評価

本研究で開発した muphic-online はシステム全体として要求と仕様を概ね満たしているが，真に児童にとって価値のあるシステムであるかどうかを評価出来ていない．したがって，本研究で開発した muphic-online を実際に児童に遊んでもらうユーザビリティテストを実施する必要がある．ユーザビリティの評価については共同研究者の高橋（敬称略）の専門領域であるため，高橋の主導の下ユーザビリティテストを実施し，その結果をシステムの設計にフィードバックしなければならない．

## 9.2. タブレット端末への対応

パソコン端末と比べて，タブレット端末はタップによる直感的な操作でアプリケーションを楽しめるため，児童との親和性は高いと推測出来る．したがって，muphic-online をタブレット端末に対応させることによって，より一層児童の興味を惹くアプリケーションを実現出来るのではないだろうか．

## 9.3. 音楽理論習得に関する検証

本研究では「表現力や創造力を養えるツール」として muphic-online を開発したが，そもそも「表現力」や「創造力」という言葉を明確に定義することは難しく，「表現力や創造力を養うことが出来たのか」という検証はさらに困難であると考えられる．表現力や創造力は曖昧で人それぞれによって違う感性であり，定量的に評価することが困難な概念である．

そこで，muphic-online の有用性を検証する指標として，小学校の学習指導要領レベルの音楽理論の習得が考えられる．muphic-online では簡単な作曲の機能を提供する．よって，児童が実際に muphic-online を遊ぶことにより，初歩的な音楽理論（小節や音階の概念，音色の違いなど）を習得することが出来れば，muphic-online は音楽教育のツールとして有用であることが立証出来ると考えられる．

## 9.4. コンピュータを用いた教育に関する検証

近年では IT 化が進み，小学校でもコンピュータを用いた授業を実施して児童の IT 社会への適応を促す必要があると考えられる．そのような場合，児童が自ら興味を持ってコンピュータに触れる環境を提供することが重要である．ここに，muphic-online の導入容易性を適用出来るのではないかと考えられる．

muphic-online はインストールが不要であるため，教員が各端末のセットアップなどを行う必要がない．よって，環境構築にかかる手間を大幅に削減出来る．そして，muphic-online

を児童が楽しんで遊べるアプリケーションとして実装し，児童が muphic-online を再び遊びたいと思えるようになれば，自ら興味を持ってコンピュータに触れる機会が多くなると考えられる. これにより, コンピュータを用いた教育や授業への貢献という意味で, muphic-online の有用性を立証出来ると考えられる.

## 10. おわりに

本研究では，児童が楽しみながら創造力や表現力を養うことが出来る Web アプリケーションとして，「メロディ付き物語創作支援 Web アプリケーション（muphic-online）」を開発した．最初にシステムの要求分析を行い，システムの仕様を定めた．そして，定めた仕様を満たすための機能を洗い出し，それらを実装可能な要素技術を用いて実装を行った．最後に，開発したシステムについて仕様と要求の満足度を考察し，今後実践すべきことをまとめた．結果として，物語作成と作曲を行うことが出来，自分の成果物の読み出しと保存が可能なシステムを開発することが出来た．

最後に，システムの運用に関して結論をまとめる．本研究で開発した muphic-online は，要求と仕様を概ね満たしていると考察出来たものの，現段階の完成度ではまだ小学校や自宅での運用は困難であると結論づける．その理由としては，物語作成と作曲を行うための最低限の機能が実装されているものの，機能の充実度や利便性に乏しく，情操教育のツールとしては準備不足であると考えられるためである．例として，muphic-online を音楽教育のツールとして使用する場合は，音楽の楽しさを児童が学べる必要がある．そのためには，豊富なバリエーションの楽器（音色）を機能として用意するといった工夫が必要である．児童に対する情操教育を実現するためには，前提として「児童が楽しみながら使用し，また遊びたいと思えるようなシステム」を開発しなければならない．よって，機能の充実や利便性の向上を図り，より一層児童の興味・関心を惹くようなシステムとして muphic-online を作り込んでいく必要がある．muphic-online の改良にはユーザビリティテストの結果を活かすことが出来るため，システム開発とユーザビリティテストを繰り返すサイクルを実現することが望ましい．

## 協同研究者の紹介

本研究を行うにあたって，1 年間共に muphic-online の開発に取り組んだ力武研究室所属の共同研究者を紹介しておく．

- 情報電子システム工学専攻　　1 年　高橋綾（敬称略）
- 情報工学科　　5 年　伊藤慶二朗
- 情報工学科　　5 年　齋藤新之助

## 謝辞

本研究を進めるにあたり，研究や卒業論文執筆に関してご指導を頂きました同研究室の力武克彰助教授に感謝致します．また，共同研究者として 1 年間研究に取り組み，親身になって研究の面倒を見て頂いた情報電子システム工学専攻の高橋綾氏に感謝致します．そして，研究に関して鋭い指摘やアドバイスを下さいました同研究室の皆様と，実験等にご協力頂いた方々に感謝致します．

## 参考文献


[1] 亀谷学人，力武克彰，佐藤貴之「児童対象メロディ付き物語創作支援システムの開発」『情報処理学会創立 50 周年記念（第 72 回）全国大会講演論文集』4 号，pp.595，2010

[2] 佐藤貴之，石澤慶子 他「児童対象メロディ付き物語創作支援システムの開発」『情報科学技術フォーラム講演論文集』7 号，pp.73，2008

[3] 大川景太，知育アプリケーションの高可用化に関する研究，卒業研究論文，pp.2，2013

[4] Jonathan Rasmusson,『アジャイルサムライー達人開発者への道』, オーム社, pp.116, 2011

[5] 佐藤貴之，石澤慶子 他「児童対象メロディ付き物語創作支援システムの開発」『情報科学技術フォーラム講演論文集』7 号，pp.71，2008

[6] 佐藤貴之，石澤慶子 他「児童対象メロディ付き物語創作支援システムの開発」『情報科学技術フォーラム講演論文集』7 号，pp.71，2008

[7] enchant.jsーA simple JavaScript framework for creating games and apps <http://enchantjs.com/ja/>（2014/02/13 アクセス）

[8] Top 5 Desktop, Tablet & Console Browsers in Japan from 1 Jan 2013 to 1 Jan 2014 <http://gs.statcounter.com/#browser-JP-daily-20130101-20140101>（2014/02/13 アクセス）

[9] node.js <http://nodejs.org/>（2014/02/13 アクセス）

[10] Socket.IO: the cross-browser WebSocket for realtime apps <http://socket.io/>（2014/02/13 アクセス）

[11] MongoDB <http://www.mongodb.org/>（2014/02/13 アクセス）

[12] Heroku | Cloud Application Platform <https://id.heroku.com/login>（2014/02/13 アクセス）