Wexibision

VRを利用した作品展示Webサイトの制作

人間情報デザインコース

1711020 川股翔太

提出日：　2020年12月15日

要旨

本研究が行われた2020年は新型コロナウイルスの世界的流行により、多くのイベントが中止となった。現地でイベントを開催する代わりとしてバーチャルリアリティを利用してインターネット上でイベントを開催することに注目が集まった。本研究の目的は、現地での展示会開催が困難な状況でも、誰もが自身の作品の展示会を開催し、誰もがさまざまな作品に触れることができる環境を提供することである。そこで、3Dやバーチャルリアリティの専門知識やコーディングが不要なバーチャル展示用Webアプリ「Wexibision」を制作した。

本研究を開始するにあたり、本論文執筆までに配信されているバーチャルリアリティを利用したコンテンツを調査した。ブラウザのみでバーチャルリアリティ体験可能なコンテンツもある一方、専用のアプリケーションが必要なサービスもある。また、ユーザーがオリジナルのバーチャル空間を作成できるサービスがあるが、専門的なソフトウェアを使用する必要がある。このように、バーチャルリアリティに関する専門的な知識が無いユーザーが気軽にバーチャルリアリティコンテンツを制作することは難しいことが考察できた。

そこで、本研究の目的を達成するため、難解な専門用語を使用せず、ボタンのクリックだけでバーチャル展示を行うことができるWebアプリケーション「Wexibision」を制作した。Wexibisionは、経済的な理由や2020年の新型コロナウイルス流行のような社会的な理由などで展示会が開催できないような場合、オンライン上で展示会を開催することができるものである。また、Webページにただ画像を貼り付けるだけではなく、額に入れた作品を壁に飾った状態で見て欲しいという要求に応えることができる

本研究では作品画像のアップロード機能とバーチャル空間表示機能という2つの基本機能を実装した。将来的に「作品を鑑賞する人と作品を展示する人を繋ぐコミュニケーションツール」としての活用を考えていることから、複数人による同時ログインやチャット機能など、まだ実装するべき機能がある。また、実際のサービス運用のためにはサーバーの安定性やセキュリティの問題など解決すべき課題がある。

目次

第1章　はじめに

1-1　本研究の背景

本研究が行われた2020年は新型コロナウイルスの世界的流行によって人々の行動が大きく変化した。2020年4月には政府より緊急事態宣言が発出され、外出自粛の要請によって多くの人が集まるイベントが軒並み中止となる異例の事態となった。2020年5月末の緊急事態宣言解除以降、徐々に制限緩和の兆しが見えつつあるものの、札幌国際芸術祭やさっぽろ雪まつりの現地開催が中止となるなど、新型コロナウイルス流行以前のような状況には未だ戻っていない状況である。

前例の無い状況の中、新たなイベントの開催方法を模索する動きが加速した。中でも「バーチャル渋谷」[1]を初めとするバーチャルリアリティを利用したコンテンツが多く配信された。

1-2　本研究の目的

上記のように、これまでオフラインで実施されていた展示やイベントが中止になっている一方で、バーチャル渋谷やオンラインさっぽろ雪まつり[2]など、オンライン上のイベントが行われるようになっている。しかしながら、個人でこのようなオンラインイベントを行おうとすると、Webサイト構築や3D、バーチャルリアリティといった専門知識が必要となる。そこで、本研究では3Dやバーチャルリアリティの専門知識やコーディングが不要なバーチャル展示用Webアプリを制作する。本研究では、現地での展示会開催が困難な状況でも、誰もが自身の作品の展示会を開催し、誰もがさまざまな作品に触れることができる環境を提供することを目的とする。

1-3　本論文の構成

本論文の構成は以下の通りである。第2章では本研究に取り掛かるにあたって事前に調査した内容について述べる。第3章では本研究で制作する「Wexibision」の提案内容について述べる。第4章では本研究「Wexibision」の制作過程について述べる。第5章では本研究についてのまとめと今後の展望について述べる。

第2章　調査

2-1　A-Frameについて

「A-Frame」[3]とは、バーチャルリアリティをWeb上で実現するためのオープンソースWebフレームワークである。Webにおけるバーチャルリアリティコンテンツの制作には、一般に「WebGL」[4]と呼ばれる仕組みを使わなければならないが、専門的な知識がなければ扱いづらいため、WebGLを簡単に利用できるように「Three.js」[5]というJavaScriptライブラリが開発されている。A-FrameはこのThree.jsをベースに動作するフレームワークであり、HTMLタグを用いてJavaScriptのプログラムを記述することなくバーチャルリアリティコンテンツを制作することができるという特徴がある。もともとA-Frameはオープンソースブラウザ FireFoxなどの開発で知られるMozilla[6]によって開発が進められ、2015年12月17日に初版となるバージョン0.1.0が一般に公開された。本論文執筆時のバージョンは1.1.0である。

2-2　類似の事例について

2020年には新型コロナウイルスの流行に対応するため、様々なイベントがバーチャルリアリティを用いて行われた。主な事例を以下に挙げる。

2-2-1　国立科学博物館「かはくVR」

「かはくVR」[7]とは、2020年5月に一般財団法人VR革新機構の協力のもと、国立科学博物館が無料で公開した3Dビューとバーチャルリアリティを利用したコンテンツである。国立科学博物館内の展示ルームを360度撮影することで制作されており、自宅からでも展示を楽しめるというコンテンツである（図1）。

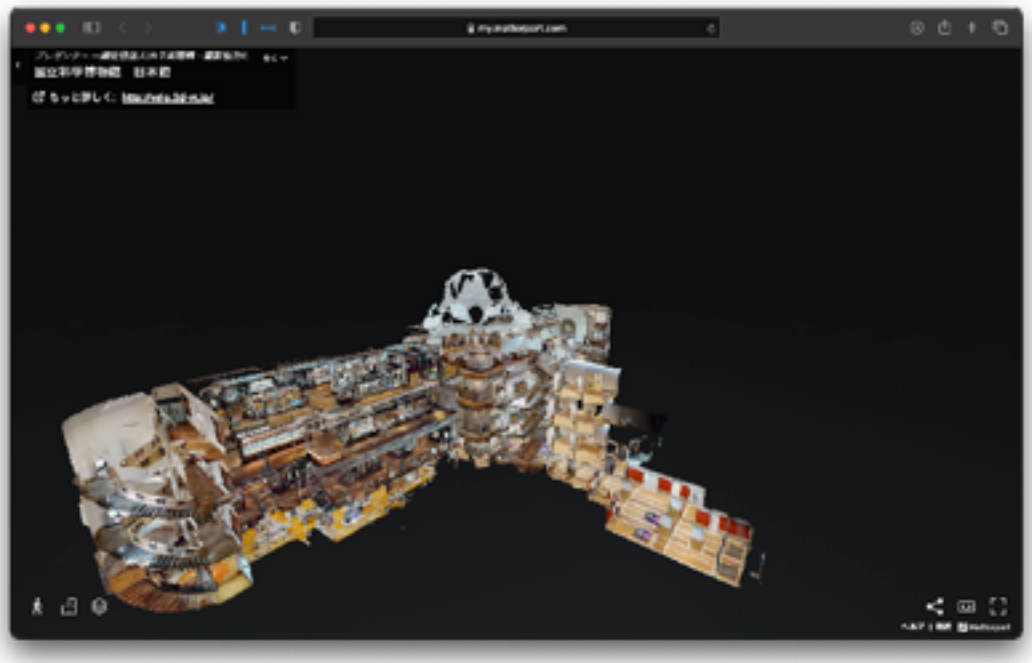

図１　かはくVR「日本館」のページ [6]

2-2-2　バーチャル渋谷

「バーチャル渋谷」[1]とは、KDDI株式会社、一般社団法人渋谷未来デザイン、一般財団法人渋谷区観光協会を中心とする「渋谷5Gエンターテイメントプロジェクト」が、2020年5月から提供している配信プラットフォームである。渋谷区を代表するスクランブル交差点周辺がバーチャル空間上に再現されており、ユーザーはアバターを用いてバーチャル空間上を自由に動き回ることができる。XRアートなど、現実の渋谷の街で提供されているコンテンツと同一のものが体験可能で、実際の渋谷の街と連携する「デジタルツイン」がコンセプトとなっている。2020年10月末には渋谷で有名なハロウィーンイベントがバーチャル渋谷でも行われた（図2）。

図2　バーチャル渋谷Webページ [1]

2-2-3　バーチャルSNS「cluster」

「cluster」[8]とは、クラスター株式会社が運営するサービスである。ユーザーはあらかじめ用意されたバーチャル空間またはゲームエンジン「Unity」[9]と、専用の開発キット「Cluster Creator Kit」を用いて作成したオリジナルのバーチャル空間で、イベントを開催することができる。プリセットとしてあらかじめ用意されているバーチャル空間には画像や動画などを映し出すことが可能なスクリーンが 1 枚設置されている。バーチャルイベントに参加するためには専用のソフトウェアやアプリをインストールする必要がある（図3）。

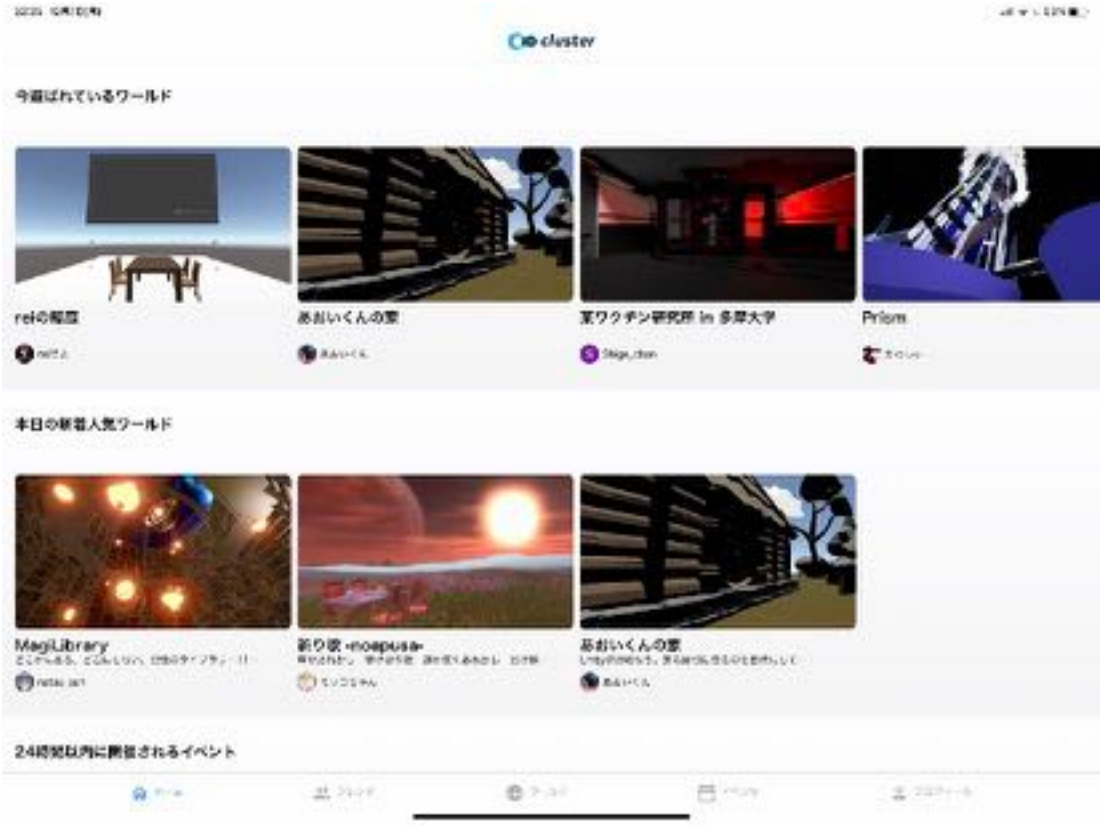

図3　iPad版「cluster」専用アプリ画面

2-3　まとめ

2020年は新型コロナウイルス流行の影響で「リアル」で行うイベントを中止せざるを得ない状況が続いたため、バーチャルリアリティを活用したイベントが多く行われた。「かはくVR」のようにブラウザのみで体験可能なコンテンツもあるが、多くはバーチャルリアリティ体験に専用のソフトウェアやアプリを必要としている。ユーザー自身の手によってバーチャル空間やバーチャルイベントを作成、開催することができるサービスがある一方、プリセットとして用意されているバーチャル空間は画像を表示できるスクリーンの数が少なく、展示会場として利用するには不向きである。バーチャルSNS「cluster」では卒業展示が行われた例もあるが、オリジナルのバーチャル空間として作成されている。作品の展示に適した環境を構築するためにはUnityというソフトウェアの知識が必要である。

第３章　「Wexibision」の提案

3-1　コンセプト

Wexibisionは「Unity」を初めとする専門ソフトウェアや「WebGL」や「Three.js」といったバーチャルリアリティに関連する専門知識が無くても、画像をアップロードするだけでバーチャル展示が行えるWebアプリケーションである。「A-Frame」でWebVRコンテンツを作成する際に必要なコーディングも不要であるため、画像を用意できるユーザーであれば誰でもバーチャル展示コンテンツを制作できる。

Wexibisionという名前は「Web」と展示を意味する英語の「Exhibition」の組み合わせから想起した造語であり、読みやすさと音の響きを考慮して名付けた。

ユーザーが操作できる項目は展示会場の名前や作品情報の入力フォームのみとなっており、Wexibision内には難解な専門用語は一切含まれていない。クリックやドラッグ、タップなどPCやスマートフォンの基本的な操作が行えるユーザーならば理解できるような簡素なインターフェースとなるように努めた。

Wexibisionの動作には特別なアプリは必要無く、一般的なブラウザのみでバーチャルリアリティ体験をすることができる。OSやハードウェアの制限も無いため、通常のWebサイトを閲覧する感覚で手軽にバーチャル鑑賞を行うことができる。

3-2　ターゲット

Wexibisionのターゲットユーザーは「何らかの理由により展示会を開催することができない人」である。Wexibisionは、経済的な理由や2020年の新型コロナウイルス流行のような社会的な理由などで展示会が開催できないような場合、オンライン上で展示会を開催することができる。また、Webページにただ画像を貼り付けるだけではなく、額に入れた作品を壁に飾った状態で見て欲しいという要求に応えることができる。

第4章　「Wexibision」の制作

4-1　制作環境

4-1-1　作業環境

Wexibisionの制作には「macOS　バージョン11.0.1　Beta」をインストールしたMacを使用し、テキストエディタに「Microsoft　VisualStudio　Code」[10]を使用した。図4は開発中の画面のスクリーンショットである。

Wexibisionの制作ではオープンソースで提供されている以下のライブラリを使用している。

- gon [11]
- CarrierWave [12]
- A-Frame
- aframe-extras [13]
- aframe-text-geometry-component [14]
- jQuery

これらのライブラリを使用しているが、Wexibisionの機能に必要なコードは全て著者自身の手によってコーディングした。

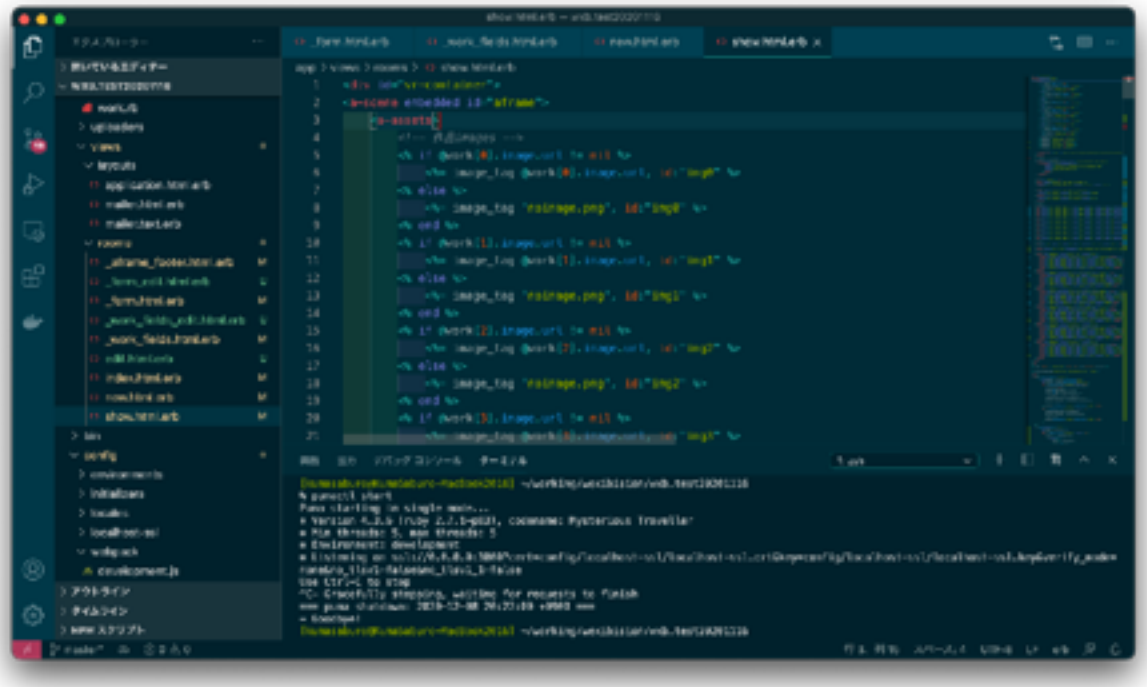

図4　開発画面のスクリーンショット

4-1-2　Ruby on Rails

Wexibisionではサーバー側のシステムに「Ruby on Rails」[15]を採用した。Wexibisionはユーザーからアップロードされた作品の画像データを保存し、指定されたページで表示するという処理が必要であり、これを実現するためにはデータベースが必要である。Ruby on Railsはデータベースと密接に連携しているため、Wexibisionに必要な処理が行えると判断し採用した。Wexibisionの制作に使用したRuby on Railsのバージョンは「6.0.3.4」である。

Ruby on Railsの動作にはRubyのインストールも必要である。Wexibisionの制作に使用したRubyのバージョンは「2.7.1」である。

4-1-3　PostgreSQL

Ruby on Railsの動作にはデータベースが必須となっている。Wexibisionでは「PostgreSQL」[16]というデータベースを利用している。インストールしたPostgreSQLのバージョンは「13.0」である。

4-1-4　使用したGem

Ruby on Railsでは、「Gem」[17]と呼ばれるプログラミング言語Rubyのパッケージ管理システムを利用して、特定の機能を容易に利用できるようにしたライブラリを使用することができる。Wexibisionの制作に使用したGemは「gon」と「CarrierWave」の２つである。

「gon」はRuby on Rails側で定義されている変数をブラウザ側のJavaScriptでも読み取ることができるようにするGemである。使用したバージョンは「6.4.0」である。

「CarrierWave」はデータのアップロードやアップロードされたデータをデータベースに格納する機能をもつGemである。CarrierWaveを使用することでクライアント側とサーバー側のデータのやり取りを容易に実装することが可能となった。使用したバージョンは「2.1.0」である。

4-1-5　使用したJavaScriptライブラリ

Ruby on Railsのバージョン6以降ではJavaScriptライブラリをパッケージ管理システム「Yarn」[18]を用いてインストールすることが一般的である。そのため、Wexibisionで利用するJavaScriptライブラリは「Yarn」を用いてインストールした。Wexibisionで使用しているJavaScriptライブラリは以下の４つである。

- A-Frame
- aframe-extras
- aframe-text-geometry-component
- jQuery

「A-Frame」はWexibisionの動作の要となるJavaScriptパッケージである。使用したバージョンは「1.0.4」である。

「aframe-extras」はA-Frameの追加機能を集めたライブラリである。主にバーチャル空間でのカメラの制御に利用している。使用したバージョンは「6.1.1」である。

「aframe-text-geometry-component」はA-Frameのタグの一つである「a-text」などでバーチャル空間上に表示するテキストを立体的にする機能を提供するライブラリである。

Wexibisionではバーチャル展示ルーム入り口にある「Wexibision」の文字に適用している。使用したバージョンは「0.5.1」である。

「jQuery」はJavaScriptでHTMLの操作を簡単に行うことができるようにするライブラリである。使用したバージョンは「3.5.1」である。

4-1-6　SSL通信

Wexibisionで利用しているA-FrameはSSL（Secure Socket Layer）による暗号化通信が必須となっている。SSL通信を行うためには「証明書」が必要である。今回はmacOSの「キーチェーンアクセス.app」で発行した証明書を利用した。PEM形式で書き出した証明書とサーバーの秘密鍵、中間認証局の証明書をRailsプロジェクトルートにある「config」ディレクトリに配置し、Ruby on Railsで利用するアプリケーションサーバー「Puma」の設定を行った。

4-1-7　システム構成図

Wexibisionのシステム構成図は図5のようになっている。

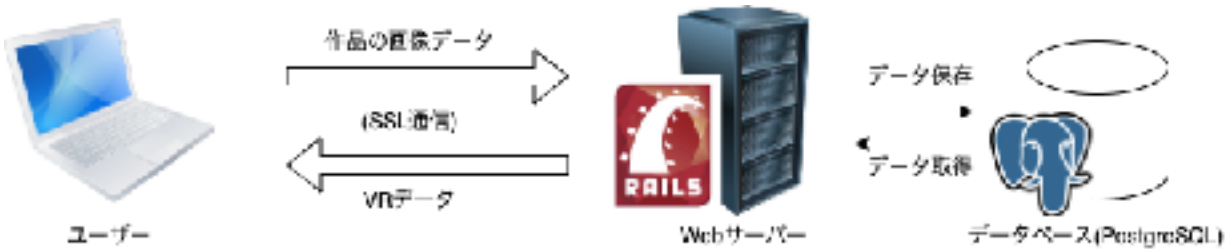

図5 Wexibisionのシステム構成図

4-2　アップロード機能

4-2-1　CarrierWave

Wexibisionの作品データアップロード機能部分にはRuby on Railsのgem「CarrierWave」[14]を利用した。

4-2-2　モデルの作成

アップロードされたデータを保存するためにはデータベースを利用する。Ruby on Railsでは「モデル」を使ってデータベースのデータにアクセスする。Wexibisionでは展示ルームの情報を格納するための「Room」と作品の情報を格納するための「Work」という２つのモデルが必要である。モデルを作成する前に、データベースにあるデータがどのように関連付けられているのか示す「ER図」を作成した（図6）。Ruby on Railsではモデルを作成すると同時に、データベースにデータを格納するための入れ物であるテーブルの設計図となる「マイグレーションファイル」が作成される。この図を元にマイグレーションファイルを編集し、データベースにテーブルとテーブルの中身であるカラムを作成した。

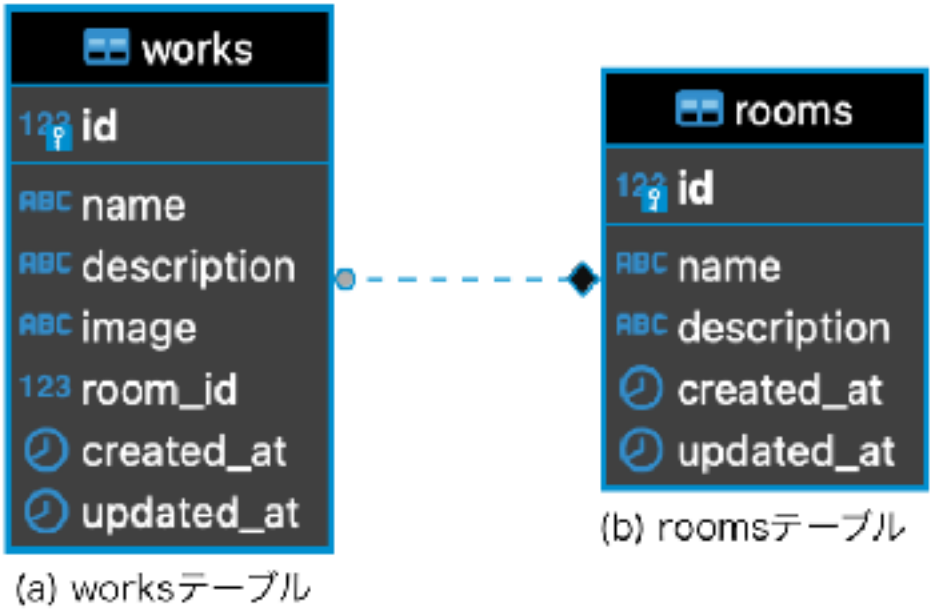

図6　実際に使用したデータベースのER図

roomsテーブルには展示ルームの名前を格納するための「name」カラムと説明文を格納するための「description」カラムがあり、worksテーブルには同じく「name」「description」カラムの他に作品の画像を格納するための「image」カラムと内部処理で使用する「room_id」がある。「created_at」と「updated_at」はそれぞれ作成、更新した日付が格納される。

4-2-3　アップロードページの制作

ユーザーが作品をアップロードする際に訪れるページは図7のように3つのパーツで構成されている。

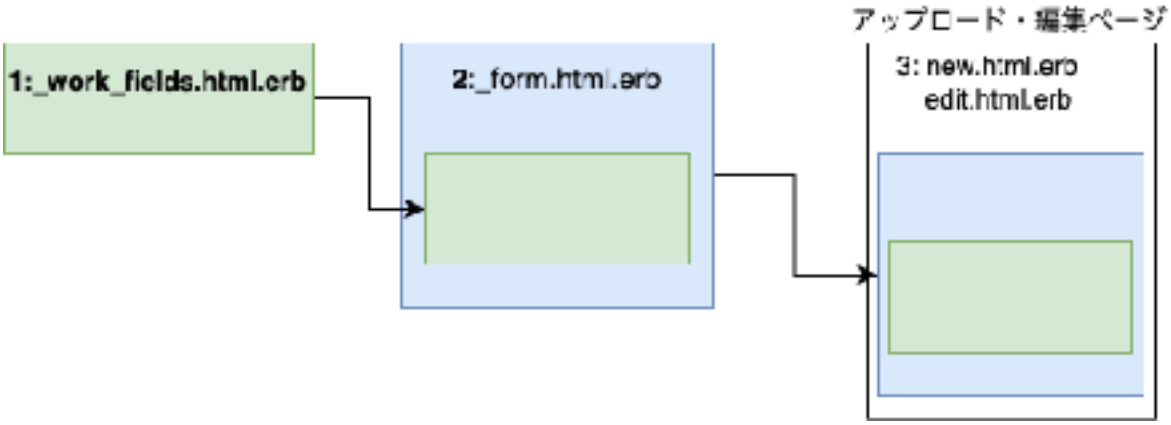

図7　アップロード・編集ページファイル構成

1つ目のパーツは「_work_fields.html.erb」である。「_work_fields.html.erb」にはバーチャル展示ルームに展示したい作品のタイトルと解説を入力するフォームと、作品の画像データをアップロードするフォームが設置されている。

2つ目のパーツは「_form.html.erb」である。「_form.html.erb」にはバーチャル展示ルームの名前と説明を入力するフォームが設置されており、コード内の「<%= render 'work_fields', f: work %>」という部分で、1つ目のパーツである「_work_fields.html.erb」を呼び出している。

3つ目のパーツは「new.html.erb」と「edit.html.erb」である。「new.html.erb」は新規にバーチャル展示ルームを作成する場合に使われるパーツで、「edit.html.erb」は既存のバーチャル展示ルームを編集する際に使われるパーツである。「new.html.erb」と「edit.html.erb」にはページの機能を明示するための「Create New Room」と「Edit Room」というページタイトルと、2つ目のパーツである「_form.html.erb」を呼び出すための「<%= render :partial => "form" %>」が記述されている。

バーチャル展示ルームの作成、及び作品データのアップロードに使うページとバーチャル展示ルーム、作品データの編集に使うページでは、ほとんど同じパーツを利用する。こ

れらのページで共通する部分を「_form.html.erb」「_work_fields.html.erb」として切り分けることで、重複する記述の無い分かりやすいコードとすることができた。

実際に制作したアップロード用のページが図8、編集用のページが図9である。

図8　アップロードページ（new.html.erb）

図9　編集用ページ（edit.html.erb）

4-3　バーチャル展示ルームの制作

4-3-1　A-Frame

バーチャル展示ルームはA-Frame[2]を用いて制作した。制作したバーチャル展示ルームの平面図は図10のようになっている。床のオブジェクトにフローリングのテクスチャを、壁のオブジェクトにはキャンバス生地のテクスチャを設定した。ライティングについても自然な様子に見えるように動作が重くなりすぎないように配慮しながら調整した。現実の展示ルームの質感に近づけることで、実際の展示の代わりとした際にも違和感がないように配慮した。カメラの中央部にはカーソルの代わりとなる丸い円を表示している。作品画像の上にカーソルが乗った時には作品鑑賞の支障とならないようにカーソルが非表示となるようになっている（図11）。

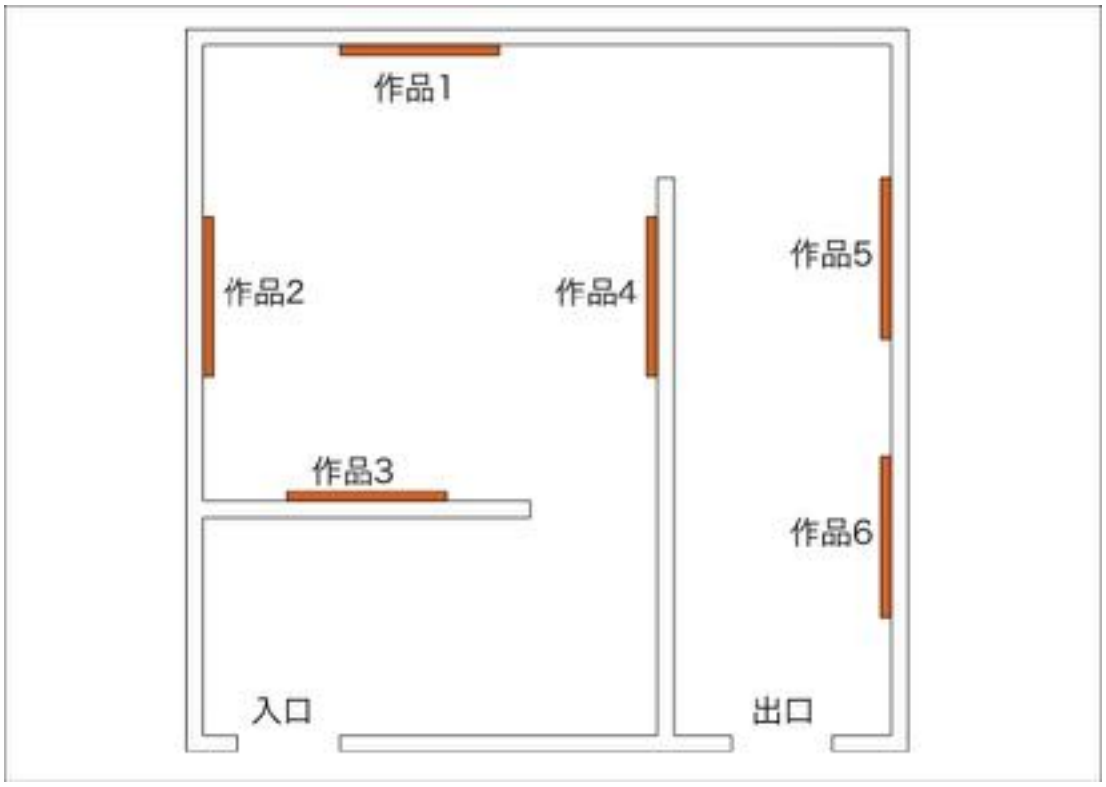

図10　バーチャル展示ルーム平面図

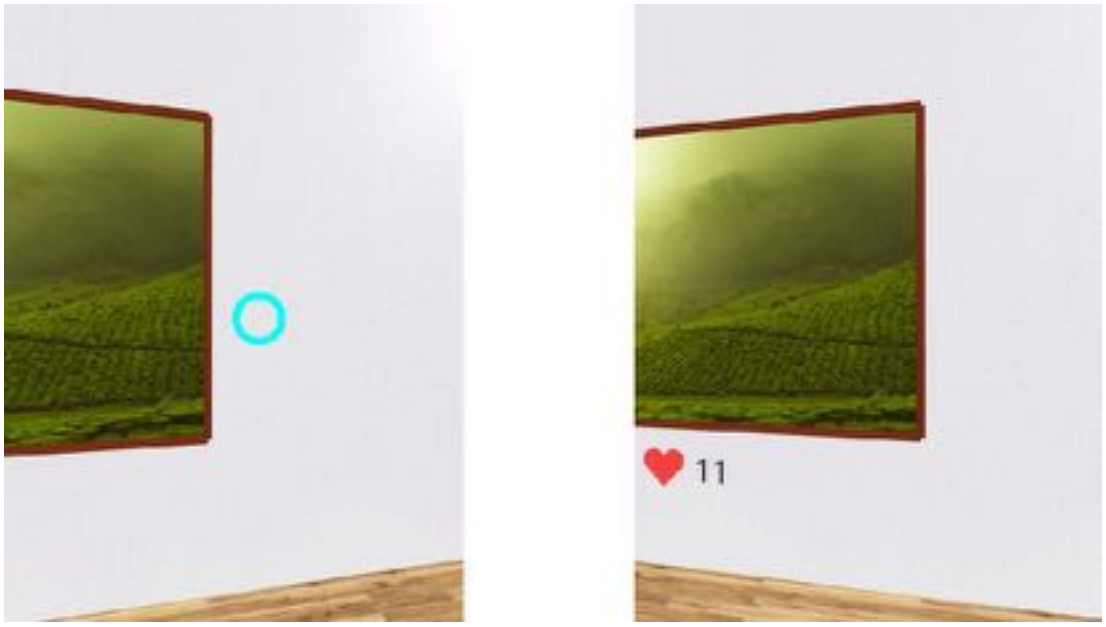

図11　カーソル（左）が作品の上に乗ると非表示になる（右）

4-3-2　作品情報の表示機能

2020年11月現在、A-Frameでは日本語のフォントをバーチャル空間上に表示することができない。この問題を解決するため、作品の名前や解説を表示する部分は通常のHTMLを使用し、VR表示部の外に表示できるようにした（図12）。

図12　解説の表示

A-FrameではJavaScriptをA-Frameコンポーネントとして登録することで、A-Frameオブジェクトと JavaScriptを連動させることができる。JavaScriptは作品画像前に設置した非表示のイベントリスナーオブジェクトの上にカーソルが乗った時に実行されるようになっている。イベントリスナーオブジェクトにはRuby on Rails側で作品に割り振られるIDと同じIDが設定されている。Javascriptではカーソルが乗ったイベントリスナーオブジェクトのIDを取得しRuby on RailsのGem「gon」を経由して取得したIDの作品情報を呼び出している。その後、取得した作品情報でHTMLを置き換えるという動作を行っている。カーソルがイベントリスナーオブジェクトから離れた時にはバーチャル展示ルームの名前と説明に置き換えるようになっている。図13は一連の処理内容を示したものである。

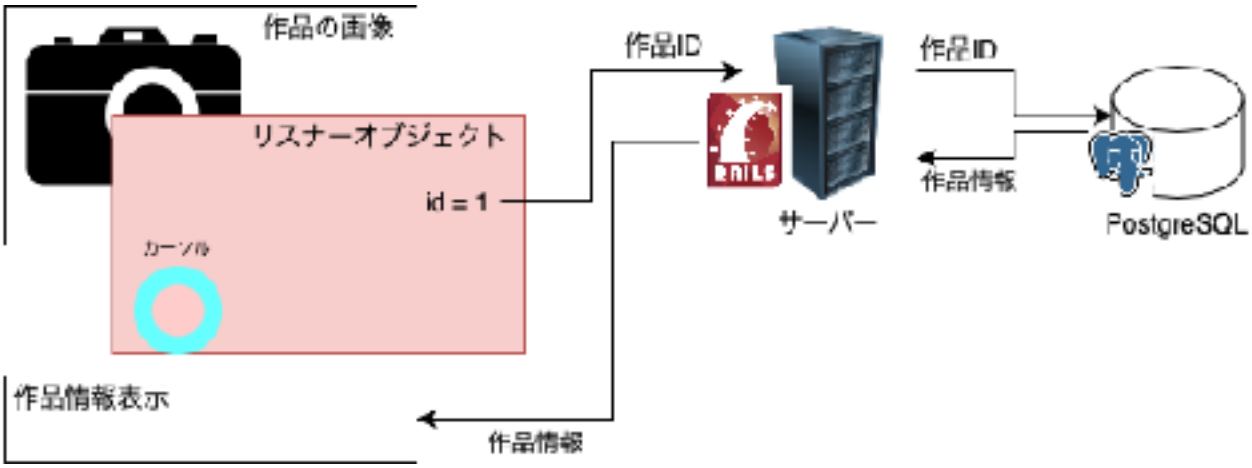

図13　作品情報表示機能の処理図

4-3-3　作品の注視時間計測機能

Twitterをはじめとするsnsには興味や賛同を示すための「いいね」という機能があり、ボタンを押すことで他人の投稿に「いいね」を送ることができるが、バーチャル空間ではボタンを押すために視点を動かす必要がある。VRゴーグルを使用しない環境では視点を動かすことがスムーズではない。従って、バーチャル空間上にボタンを配置するのはユーザビリティに欠けると判断した。また、作品情報を表示する画面下部のHTML部分にボタンを表示することもバーチャル空間と画面下部とのカーソル移動が必要なため不便である。このため、WexibisionではJavaScriptを利用して作品を注視していた時間を計測することで、ユーザーがどれだけその作品に興味関心を持っているかについての指標とすることとした。この注視時間は既存のSNSにおける「いいね」と同じ意味を持っている。展示されている作品にどれだけユーザーが興味関心を示しているかを、他のユーザーや作品の作者と共有することを目的としている。

この機能はJavaScriptによって実装しており、イベントリスナーオブジェクトの上にカーソルが乗った時にタイマーがスタートするようにプログラムしている。注視時間は作品の下にある数字で表示される。「setAttribute」メソッドを使用して「a-text」タグの「value」属性を変更するという処理によって実現している（図14）。

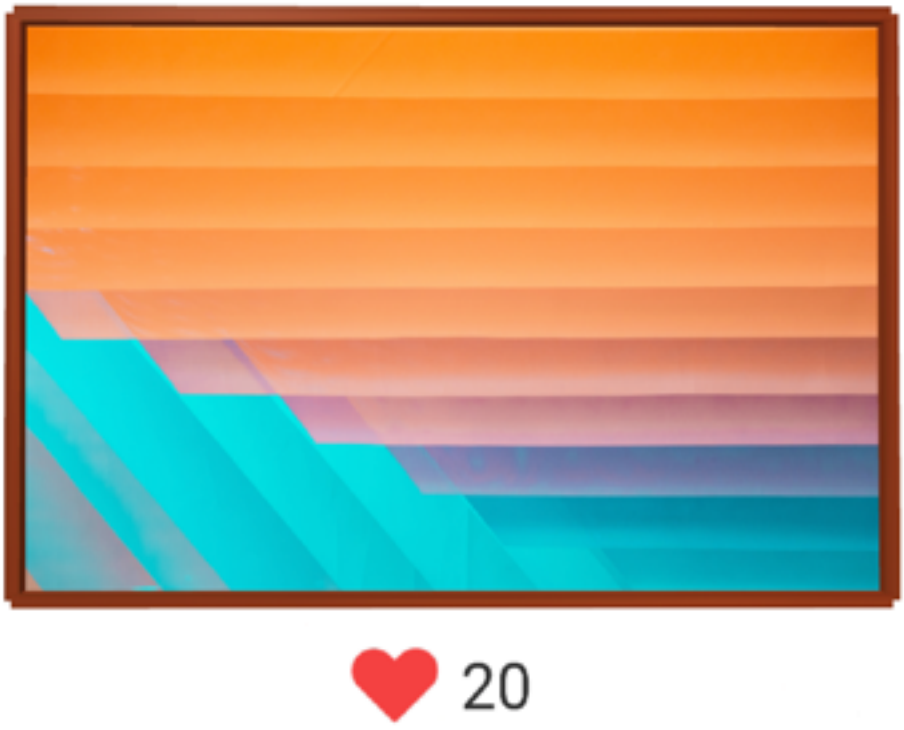

図14　作品の下にある注視時間を示す数字

4-4　制作物の解説

4-4-1　アップロード機能の解説

ユーザーはアップロードページ（図8）にアクセスすることでバーチャル展示ルームの登録と作品のアップロードを行うことができる。アップロードできる作品の数は最大6個で、各作品にタイトルと解説をつけることができる。作品は必ず６個アップロードしなければならないというわけではなく、空白のまま送信することも可能である。タイトルや解説の入力を終えた後はページ最下部にある「Submit」ボタンを押すことでバーチャル展示ルームを作成することができる。「Submit」ボタンを押した後はバーチャル展示ルームを一覧できるインデックスページにリダイレクトされる（図15）。

図15　インデックスページ

インデックスページではバーチャル展示ルームへのリンク、編集、削除のためのボタンが設置されている。「見る」ボタンを押すとウィンドウが現在のタブと別のタブで開き、バーチャル展示ルームを見ることができる。「編集」ボタンを押すとバーチャル展示ルームの編集ページにリンクし、タイトルや解説の変更や作品画像の編集を行うことができる。「削除」ボタンを押すと確認のポップアップが表示され、OKを押すとバーチャル展示ルームを削除することができる。インデックスページ最下部には「新規作成」ボタンが

あり、アップロードページにリンクすることができる。図16はユーザー操作の流れを示したフローチャートである。

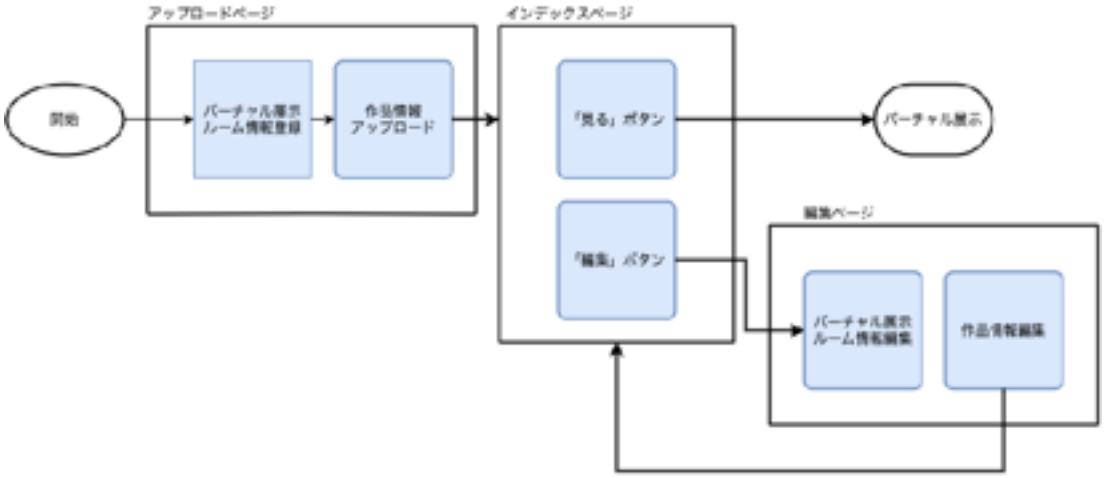

図16　ユーザー操作のフローチャート

4-4-2　バーチャル展示ルームの解説

バーチャル展示ルームのページレイアウトは図17のようになっている。画面上部の広範囲を占める部分がバーチャル空間である。画面下部の帯のような部分ではバーチャル展示ルームや作品のタイトル、解説が表示される。解説はバーチャルリアリティ体験を妨げることがないよう、初期状態では非表示となっており、「もっと見る」ボタンを押すことで前述の図12のように解説がドロップダウンする。

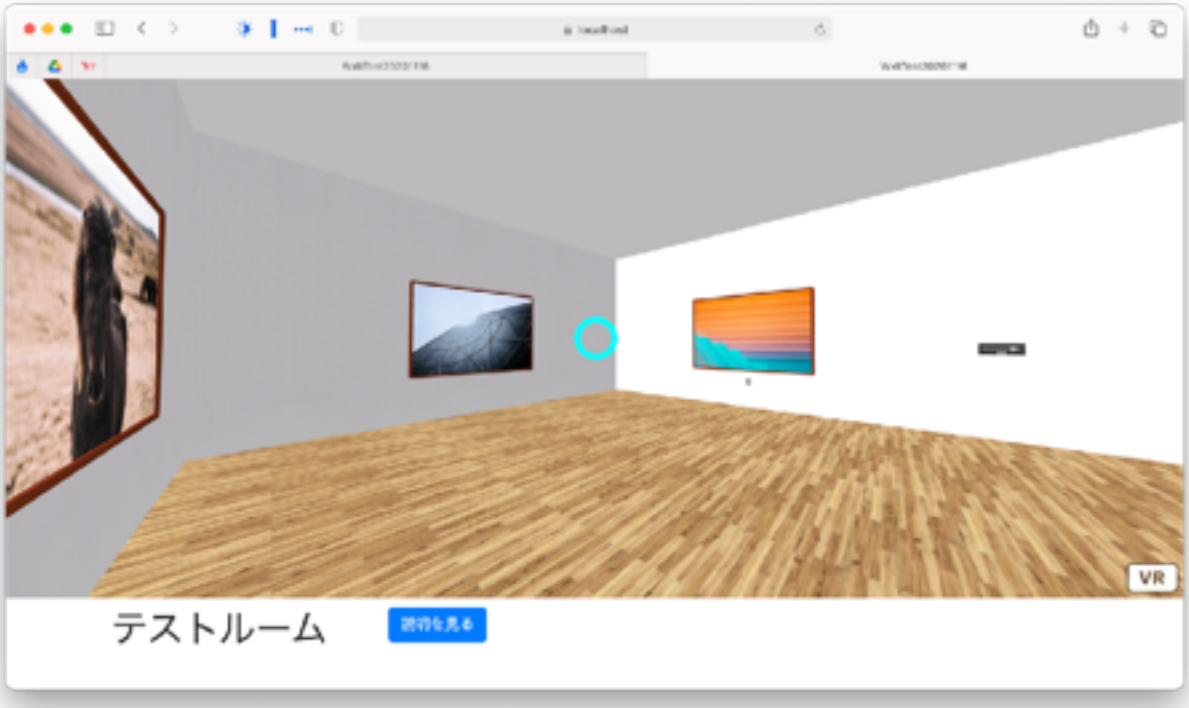

図17　バーチャル展示ルーム

バーチャル展示ルームではカメラを移動して自由に空間内を歩き回ることができる。PCでは「WASD」キーを使って移動できる。各キーはそれぞれ「W」キーで前、「S」キーで後、「A」キーで左、「D」キーで右に進むように割り当てられている。

スマートフォンやタブレットでは1本指タップで前進、2本指タップで後進できる。左右にスワイプすることで移動方向を変えることができる。

視点の移動は、PCではマウスのドラッグ操作で、スマートフォンでは端末内蔵のジャイロを利用してスマートフォン自体を動かすことで行うことができる。

4-5　制作物の今後

今回制作したWexibisionは作品アップロード機能、バーチャルリアリティ体験機能といった基本的な機能のみが利用可能な状態である。Wexibisionは仮想空間を提供するアプリケーションであり、時間や空間の制限を超えてコミュニケーションを行うことが可能であることから、「作品を鑑賞する人と作品を展示する人を繋ぐコミュニケーションツール」としての活用を考えている。このような活用をするためには、以下のような機能の実装が必要である。

まず、複数人同時ログイン機能である。現段階では同じバーチャル空間に存在できるのは1人のユーザーのみであるが、複数人で同時ログインが可能になれば他人と一緒に作品を鑑賞できる。複数人で同時に空間に存在することができなければ、コミュニケーションを行うことは困難である。

次に、チャット機能である。複数人で同時ログインが可能となれば、チャット機能を用いて他人と会話ができる。チャット機能はインターネットにおける基本的なコミュニケーション手段の一つであり、同じ作品を鑑賞している人と意見交換をするといったコミュニティ機能をWexibisionに持たせることができる。現実の展示会場で長時間作品の前に居座って会話を続けることは他人の迷惑となるが、バーチャル空間であるWexibisionでは他人の迷惑となることはない。

次に、アバター機能である。アバターとはバーチャル空間上で表示されるユーザーの姿である。実際の例として、バーチャル渋谷ではバーチャル空間にログインする前に自身のアバターを設定する。バーチャル空間内で容易に個人を識別するためには、バーチャルな姿形が個々人で異なっている必要がある。バーチャルリアリティを利用したコミュニケーションツールにはアバター機能は必要な機能である。アバターの姿で個人を識別できるようになれば、例えば展示されている作品の作者がバーチャル空間内で発表会を行うといったことが行いやすくなる。

第5章　最後に

5-1　本研究のまとめ

2020年はバーチャルリアリティを利用したコンテンツが多く配信された。コンテンツのバーチャルリアリティを体験するためには別途アプリが必要なことがあり、スムーズな体験をすることができない。また、バーチャルリアリティを利用したコンテンツの制作には専門的な知識が必要であり、学習コストが高い。

Webベースのアプリケーションであり、展示したい作品をアップロードするだけでバーチャル展示を行える「Wexibision」を制作することで、バーチャルリアリティを利用した展示会の開催における問題点を解消する方法を提案した。

5-2　本研究の展望

本研究では「ユーザー自身が作品をアップロードし、アップロードされた作品をバーチャル空間に表示する」という基本となる機能のみを実装した。実際のサービスとして運用するためにはユーザーのログイン機能や、第4章 4-5で述べたアバター機能など必要な機能がある。また、セキュリティ面の課題や同時に多数のユーザーがサーバーに接続した際の安定性など本研究では触れていない点がいくつかある。このような課題を解決し、必要な機能を実装することでボタンをクリックするだけで簡単に利用でき、アプリのインストールが不要なバーチャル展示プラットフォームとして活用できる可能性がある。

謝辞

本研究を進めるにあたり、丁寧なご指導を頂いた大渕一博先生に感謝致します。また、さまざまなご指摘を下さったゼミの同期の方々に感謝致します。

参考文献

[1] バーチャル渋谷紹介ページ,
https://cluster.mu/w/79347fb9-05f5-429e-ab5f-8951ee8cd966 , 2020年12月3日閲覧

[2] さっぽろ雪まつり公式サイト, https://www.snowfes.com, 2021年1月11日閲覧

[3] A-Frame 公式サイト, https://aframe.io, 2020年12月8日閲覧

[4] WebGL 紹介サイト, https://www.khronos.org/webgl/ , 2020年12月8日閲覧

[5] Three.js 公式サイト, https://threejs.org , 2020年12月8日閲覧

[6] Mozilla 公式サイト, https://www.mozilla.org/ja/, 2020年12月8日閲覧

[7] かはくVR日本館ページ, https://my.matterport.com/show/?m=i7hrHfp7VE2 ,
2020年12月3日閲覧

[8] バーチャルSNS cluster 公式サイト, https://cluster.mu , 2020年12月8日閲覧

[9] Unity 公式サイト, https://unity.com/ja , 2020年12月8日閲覧

[10] Microsoft VisualStudio Code 公式ダウンロードページ,
https://azure.microsoft.com/ja-jp/products/visual-studio-code/ , 2020年12月8日閲覧

[11] Gon GitHubページ, https://github.com/gazay/gon , 2020年12月8日閲覧

[12] CarrierWave GitHubページ,
https://github.com/carrierwaveuploader/carrierwave , 2020年12月8日閲覧

[13] aframe-extras GitHubページ,
https://github.com/n5ro/aframe-extras , 2020年12月8日閲覧

[14] aframe-text-geometry-component GitHubページ,
https://github.com/supermedium/superframe/tree/master/components/text-geometry ,
2020年12月8日閲覧

[15] Ruby on Rails 公式サイト, https://rubyonrails.org , 2020年12月8日閲覧

[16] PostgreSQL 公式サイト, https://www.postgresql.org , 2020年12月8日閲覧

[17] Gem 公式サイト, https://rubygems.org , 2020年12月8日閲覧

[18] Yarn 公式サイト, https://yarnpkg.com , 2020年12月8日閲覧

付録

index.html.erb

```
<div class="container">
    <h1>Rooms Index</h1>


    <% @room.each do |room| %>
        <div class="card">
            <div class="card-header">
                <%= room.name %>
            </div>
            <div class="card-body">
                <div id="desc-container">
                    <%= room.description %>
                </div>
                <div>
                    <%= link_to '見る', room, target: :_blank, rel: "noopener noreferrer",
class: "btn btn-primary" %>
                    <%= link_to '編集', edit_room_path(room), class: "btn btn-primary" %>
                    <%= link_to '削除', room, method: :delete, data: { confirm: '
本当に削除してよろしいですか？' }, class: "btn btn-danger" %>
                </div>
            </div>
        </div>
    <% end %>
    <%= link_to '新規作成', new_room_path, class: "btn btn-primary" %>
</div>

<%= javascript_pack_tag 'styles/index', 'data-turbolinks-track': 'reload' %>
<%= stylesheet_pack_tag 'styles/index', 'data-turbolinks-track': 'reload' %>
```

new.html.erb

```
<div class="container">
    <h1>Create New Room</h1>

    <%= render :partial => "form" %>

</div>

<%= javascript_pack_tag 'styles/form', 'data-turbolinks-track': 'reload' %>
<%= stylesheet_pack_tag 'styles/form', 'data-turbolinks-track': 'reload' %>
```

edit.html.erb

```
<div class="container">
    <h1>Edit Room</h1>

    <%= render :partial => "form" %>

</div>

<%= javascript_pack_tag 'styles/form', 'data-turbolinks-track': 'reload' %>
<%= stylesheet_pack_tag 'styles/form', 'data-turbolinks-track': 'reload' %>
```

show.html.erb

```
<!-- Camera -->
<!-- ポインターロックなし -->
<a-entity position="0 1.7 3" movement-controls="constrainToNavMesh: true; speed:0.1">
    <a-entity camera look-controls wasd-controls>
        <a-entity id="cursor" position="0 0 -1"
            geometry="primitive: ring; radiusInner: 0.05; radiusOuter: 0.07;"
            material="color: cyan; shader: flat"
            cursor="maxDistance: 30; fuse: true">
        </a-entity>
    </a-entity>
</a-entity>

<!-- VR -->
<a-sky color="#0095DD"></a-sky>
<a-entity geometry="primitive: plane; width: 30; height: 30"
    material="src: #flooring; repeat: 30 10" position="7.5 0 -7.5" rotation="-90 0 0">
</a-entity>

<!-- 廊路 -->
<a-entity geometry="primitive: plane; width: 0.5; height: 0.2"
    material="src: #arrow-L;" position="9.8 1.6 -3.4" rotation="0 -90 0">
</a-entity>
<a-entity geometry="primitive: plane; width: 0.5; height: 0.2"
    material="src: #arrow-R;" position="9 1.6 -14.84" rotation="0 0 0">
</a-entity>
```

show.html.erb（つづき）

```
<div id="vr-container">
<a-scene embedded id="aframe">
    <a-assets>
        <!-- 作品images -->
        <% if @work[0].image.url != nil %>
            <%= image_tag @work[0].image.url, id:"img0" %>
        <% else %>
            <%= image_tag 'noimage.png', id:"img0" %>
        <% end %>
        <% if @work[1].image.url != nil %>
            <%= image_tag @work[1].image.url, id:"img1" %>
        <% else %>
            <%= image_tag 'noimage.png', id:"img1" %>
        <% end %>
        <% if @work[2].image.url != nil %>
            <%= image_tag @work[2].image.url, id:"img2" %>
        <% else %>
            <%= image_tag 'noimage.png', id:"img2" %>
        <% end %>
        <% if @work[3].image.url != nil %>
            <%= image_tag @work[3].image.url, id:"img3" %>
        <% else %>
            <%= image_tag 'noimage.png', id:"img3" %>
        <% end %>
        <% if @work[4].image.url != nil %>
            <%= image_tag @work[4].image.url, id:"img4" %>
        <% else %>
            <%= image_tag 'noimage.png', id:"img4" %>
        <% end %>
        <% if @work[5].image.url != nil %>
            <%= image_tag @work[5].image.url, id:"img5" %>
        <% else %>
            <%= image_tag 'noimage.png', id:"img5" %>
        <% end %>
        <!-- テクスチャー -->
        <%= image_tag 'flooring.jpg', id:"flooring" %>
        <%= image_tag 'wood.jpg', id:"wood" %>
        <%= image_tag 'canvas.jpg', id:"canvas" %>
        <%= image_tag 'arrow-R.jpg', id:"arrow-R" %>
        <%= image_tag 'arrow-L.jpg', id:"arrow-L" %>
        <%= image_tag 'hearts.png', id:"hearts" %>

        <!-- フォント -->
        <a-asset-item id="Font" src="
https://rawgit.com/ngokevin/kframe/master/components/text-geometry/lib/helvetiker_regular.typ
eface.json
"></a-asset-item>
    </a-assets>
```

show.html.erb（つづき）

```
<!--外側-->
<!--south-->
<a-box material="src: #canvas; repeat: 5 1" color="#FFF" depth="0.3" height="5" width="1"
position="0.5 2.5 0" rotation="0 0 0"></a-box>
<a-box material="src: #canvas; repeat: 5 1" color="#FFF" depth="0.3" height="2.5" width="2"
position="2 3.75 0" rotation="0 0 0"></a-box>
<a-box material="src: #canvas; repeat: 5 1" color="#FFF" depth="0.3" height="5" width="8"
position="7 2.5 0" rotation="0 0 0"></a-box>
<a-box material="src: #canvas; repeat: 5 1" color="#FFF" depth="0.3" height="5" width="1"
position="11.5 2.5 0" rotation="0 0 0"></a-box>
<a-box material="src: #canvas; repeat: 5 1" color="#FFF" depth="0.3" height="2.5" width="2"
position="13 3.75 0" rotation="0 0 0"></a-box>
<a-box material="src: #canvas; repeat: 5 1" color="#FFF" depth="0.3" height="5" width="1"
position="14.5 2.5 0" rotation="0 0 0"></a-box>
<!--west-->
<a-box material="src: #canvas; repeat: 5 1" color="#FFF" depth="0.3" height="5" width="15"
position="0 2.5 -7.5" rotation="0 -90 0"></a-box>
<!--north-->
<a-box material="src: #canvas; repeat: 5 1" color="#FFF" depth="0.3" height="5" width="15"
position="7.5 2.5 -15" rotation="0 0 0"></a-box>
<!--east-->
<a-box material="src: #canvas; repeat: 5 1" color="#FFF" depth="0.3" height="5" width="15"
position="15 2.5 -7.5" rotation="0 90 0"></a-box>
<!--内側-->
<a-box material="src: #canvas; repeat: 5 1" color="#FFF" depth="0.3" height="5" width="7"
position="3.5 2.5 -5" rotation="0 0 0"></a-box>
<a-box material="src: #canvas; repeat: 5 1" color="#FFF" depth="0.3" height="5" width="12"
position="10 2.5 -6" rotation="0 90 0"></a-box>
<!--天井-->
<a-box color="#FFF" depth="0.3" height="15" width="15" position="7.5 5 -7.5" rotation="90 0 0
"></a-box>

<!-- 入口 -->
<a-entity position="0.45 1.7 -4.7" text-geometry="
value: Wexibision; font: #Font; height: 0.2; size: 0.9" material="color: #333;"></a-entity>
```

show.html.erb（つづき）

```
<!-- 作品 -->
<a-entity class="gakubuchi" position="6.3 -0.5 -14.7" rotation="0 90 0">
    <a-image src="#img0" width=3 height=2 position="0 2.5 -1.5" rotation="0 -90 0"></a-image>
    <a-entity timer1 listener geometry="primitive: plane;" id="0" class="listener-object"
position="-0.1 2.5 -1.5" rotation="0 -90 0" scale="3 2 1" visible="false"></a-entity>
    <a-box src="#wood" depth="0.05" height="2" width="0.05" position="0 2.5 -0" rotation="
0 0 0"></a-box>
    <a-box src="#wood" depth="0.05" height="2" width="0.05" position="0 2.5 -3" rotation="
0 0 0"></a-box>
    <a-box src="#wood" depth="0.05" height="3" width="0.05" position="0 3.5 -1.5" rotation="
90 0 0"></a-box>
    <a-box src="#wood" depth="0.05" height="3" width="0.05" position="0 1.5 -1.5" rotation="
90 0 0"></a-box>
    <a-image src="#hearts" width=0.3 height=0.3 position="0 1.2 -1.7" rotation="0 -90 0"></
a-image>
    <a-text id="like1" position="0 1.2 -1.5" color="#333" value="0" rotation="0 -90 0"></
a-text>
</a-entity>
<a-entity class="gakubuchi" position="0.3 -0.5 -8" rotation="0 0 0">
    <a-image src="#img1" width=3 height=2 position="0 2.5 -1.5" rotation="0 90 0"></a-image>
    <a-entity timer2 listener material="color: red;" geometry="primitive: plane;" id="1"
class="listener-object" position="0.02 2.5 -1.5" rotation="0 90 0" scale="3 2 1" visible="
false"></a-entity>
    <a-box src="#wood" depth="0.05" height="2" width="0.05" position="0 2.5 -0" rotation="
0 0 0"></a-box>
    <a-box src="#wood" depth="0.05" height="2" width="0.05" position="0 2.5 -3" rotation="
0 0 0"></a-box>
    <a-box src="#wood" depth="0.05" height="3" width="0.05" position="0 3.5 -1.5" rotation="
90 0 0"></a-box>
    <a-box src="#wood" depth="0.05" height="3" width="0.05" position="0 1.5 -1.5" rotation="
90 0 0"></a-box>
    <a-image src="#hearts" width=0.3 height=0.3 position="0 1.2 -1.3" rotation="0 90 0"></
a-image>
    <a-text id="like2" position="0 1.2 -1.5" color="#333" value="0" rotation="0 90 0"></
a-text>
</a-entity>
```

show.html.erb（つづき）

```
<a-entity class="gakubuchi" position="5.3 -0.5 -5.2" rotation="0 90 0">
    <a-image src="#img2" width=3 height=2 position="0 2.5 -1.5" rotation="0 90 0"></a-image>
    <a-entity timer3 listener material="color: red;" geometry="primitive: plane;" id="2"
class="listener-object" position="0.05 2.5 -1.5" rotation="0 90 0" scale="3 2 1" visible="
false"></a-entity>
    <a-box src="#wood" depth="0.05" height="2" width="0.05" position="0 2.5 -0" rotation="
0 0 0"></a-box>
    <a-box src="#wood" depth="0.05" height="2" width="0.05" position="0 2.5 -3" rotation="
0 0 0"></a-box>
    <a-box src="#wood" depth="0.05" height="3" width="0.05" position="0 3.5 -1.5" rotation="
90 0 0"></a-box>
    <a-box src="#wood" depth="0.05" height="3" width="0.05" position="0 1.5 -1.5" rotation="
90 0 0"></a-box>
    <a-image src="#hearts" width=0.3 height=0.3 position="0 1.2 -1.3" rotation="0 90 0"></
a-image>
    <a-text id="like3" position="0 1.2 -1.5" color="#333" value="0" rotation="0 90 0"></
a-text>
</a-entity>
<a-entity class="gakubuchi" position="9.8 -0.5 -8" rotation="0 0 0">
    <a-image src="#img3" width=3 height=2 position="0 2.5 -1.5" rotation="0 -90 0"></a-image>
    <a-entity timer4 listener material="color: red;" geometry="primitive: plane;" id="3"
class="listener-object" position="-0.03 2.5 -1.5" rotation="0 -90 0" scale="3 2 1" visible="
false"></a-entity>
    <a-box src="#wood" depth="0.05" height="2" width="0.05" position="0 2.5 -0" rotation="
0 0 0"></a-box>
    <a-box src="#wood" depth="0.05" height="2" width="0.05" position="0 2.5 -3" rotation="
0 0 0"></a-box>
    <a-box src="#wood" depth="0.05" height="3" width="0.05" position="0 3.5 -1.5" rotation="
90 0 0"></a-box>
    <a-box src="#wood" depth="0.05" height="3" width="0.05" position="0 1.5 -1.5" rotation="
90 0 0"></a-box>
    <a-image src="#hearts" width=0.3 height=0.3 position="0 1.2 -1.7" rotation="0 -90 0"></
a-image>
    <a-text id="like4" position="0 1.2 -1.5" color="#333" value="0" rotation="0 -90 0"></
a-text>
</a-entity>
<a-entity class="gakubuchi" position="14.8 -0.5 -8.7" rotation="0 0 0">
    <a-image src="#img4" width=3 height=2 position="0 2.5 -1.5" rotation="0 -90 0"></a-image>
    <a-entity timer5 listener material="color: red;" geometry="primitive: plane;" id="4"
class="listener-object" position="-0.003 2.5 -1.5" rotation="0 -90 0" scale="3 2 1" visible="
false"></a-entity>
    <a-box src="#wood" depth="0.05" height="2" width="0.05" position="0 2.5 -0" rotation="
0 0 0"></a-box>
    <a-box src="#wood" depth="0.05" height="2" width="0.05" position="0 2.5 -3" rotation="
0 0 0"></a-box>
    <a-box src="#wood" depth="0.05" height="3" width="0.05" position="0 3.5 -1.5" rotation="
90 0 0"></a-box>
    <a-box src="#wood" depth="0.05" height="3" width="0.05" position="0 1.5 -1.5" rotation="
90 0 0"></a-box>
    <a-image src="#hearts" width=0.3 height=0.3 position="0 1.2 -1.7" rotation="0 -90 0"></
a-image>
    <a-text id="like5" position="0 1.2 -1.5" color="#333" value="0" rotation="0 -90 0"></
a-text>
</a-entity>
```

show.html.erb（つづき）

```
    <a-entity class="gakubuchi" position="14.8 -0.5 -2.9" rotation="0 0 0">
        <a-image src="#img5" width=3 height=2 position="0 2.5 -1.5" rotation="0 -90 0"></
a-image>
        <a-entity timer6 listener material="color: red;" geometry="primitive: plane;" id="5"
class="listener-object" position="-0.003 2.5 -1.5" rotation="0 -90 0" scale="3 2 1" visible="
false"></a-entity>
        <a-box src="#wood" depth="0.05" height="2" width="0.05" position="0 2.5 -0" rotation=
"0 0 0"></a-box>
        <a-box src="#wood" depth="0.05" height="2" width="0.05" position="0 2.5 -3" rotation=
"0 0 0"></a-box>
        <a-box src="#wood" depth="0.05" height="3" width="0.05" position="0 3.5 -1.5"
rotation="90 0 0"></a-box>
        <a-box src="#wood" depth="0.05" height="3" width="0.05" position="0 1.5 -1.5"
rotation="90 0 0"></a-box>
        <a-image src="#hearts" width=0.3 height=0.3 position="0 1.2 -1.7" rotation="0 -90 0"
></a-image>
        <a-text id="like6" position="0 1.2 -1.5" color="#333" value="0" rotation="0 -90 0"></
a-text>
    </a-entity>

</a-scene>
</div>

<%= render :partial => "aframe_footer", locals: {room: @room, work_info: @work_info } %>

<%= javascript_pack_tag 'aframe/aframe', 'data-turbolinks-track': 'reload' %>
<%= stylesheet_pack_tag 'aframe/aframe', 'data-turbolinks-track': 'reload' %>

 <script>
    AFRAME.registerComponent('listener', {
        init: function(){
            var name_area = document.getElementById("name");
            var desc_area = document.getElementById("description");
            this.el.addEventListener('mouseenter', function(evt){
                //カーソルが乗っているオブジェクトのidを変数に格納
                var work_id = this.id;
                console.log(work_id);
                console.log(gon.work[work_id].name);
                console.log(gon.work[work_id].description);

                //gonで取得したrailsのデータに書き換え
                name_area.textContent = gon.work[work_id].name;
                desc_area.textContent = gon.work[work_id].description;

                //カーソル非表示
                document.querySelector('#cursor').setAttribute('visible', false);
            });
            this.el.addEventListener('mouseleave', function (evt) {
                //gonで取得したrailsのデータに書き換え
                name_area.textContent = gon.room.name;
                desc_area.textContent = gon.room.description;

                //カーソル表示
                document.querySelector('#cursor').setAttribute('visible', true);
            });
        }
    });
```

show.html.erb（つづき）

```
AFRAME.registerComponent('timer1', {
    init: function () {
        var count1; //カウントの初期値
        count1 = 0;
        this.el.addEventListener('mouseenter', function (evt) {
            //count = 0; //カウントリセット
            Timer1 = setInterval(
                function(){
                    count1++; //カウントアップ
                    console.log(count1);
                    document.getElementById('like1').setAttribute('value', count1);
                },
                1000 //1000ms(1s)ごとに実行
                );
            });
        this.el.addEventListener('mouseleave', function (evt) {
            clearInterval( Timer1 );
        });
    }
});
```

```
AFRAME.registerComponent('timer2', {
    init: function () {
        var count2; //カウントの初期値
        count2 = 0;
        this.el.addEventListener('mouseenter', function (evt) {
            //count = 0; //カウントリセット
            Timer2 = setInterval(
                function(){
                    count2++; //カウントアップ
                    console.log(count2);
                    document.getElementById('like2').setAttribute('value', count2);
                },
                1000 //1000ms(1s)ごとに実行
                );
            });
        this.el.addEventListener('mouseleave', function (evt) {
            clearInterval( Timer2 );
        });
    }
});
```

show.html.erb（つづき）

```
AFRAME.registerComponent('timer3', {
    init: function () {
        var count3; //カウントの初期値
        count3 = 0;
        this.el.addEventListener('mouseenter', function (evt) {
            //count = 0; //カウントリセット
            Timer3 = setInterval(
                function(){
                    count3++; //カウントアップ
                    console.log(count3);
                    document.getElementById('like3').setAttribute('value', count3);
                },
                1000 //1000ms(1s)ごとに実行
                );
            });
        this.el.addEventListener('mouseleave', function (evt) {
            clearInterval( Timer3 );
        });
    }
});

AFRAME.registerComponent('timer4', {
     init: function () {
         var count4; //カウントの初期値
         count4 = 0;
         this.el.addEventListener('mouseenter', function (evt) {
             //count = 0; //カウントリセット
             Timer4 = setInterval(
                 function(){
                     count4++; //カウントアップ
                     console.log(count4);
                     document.getElementById('like4').setAttribute('value', count4);
                 },
                 1000 //1000ms(1s)ごとに実行
                 );
             });
         this.el.addEventListener('mouseleave', function (evt) {
             clearInterval( Timer4 );
         });
     }
 });
```

show.html.erb（つづき）

```
AFRAME.registerComponent('timer5', {
    init: function () {
        var count5; //カウントの初期値
        count5 = 0;
        this.el.addEventListener('mouseenter', function (evt) {
            //count = 0; //カウントリセット
            Timer5 = setInterval(
                function(){
                    count5++; //カウントアップ
                    console.log(count5);
                    document.getElementById('like5').setAttribute('value', count5);
                },
                1000 //1000ms(1s)ごとに実行
                );
            });
        this.el.addEventListener('mouseleave', function (evt) {
            clearInterval( Timer5 );
        });
    }
});

    AFRAME.registerComponent('timer6', {
        init: function () {
            var count6; //カウントの初期値
            count6 = 0;
            this.el.addEventListener('mouseenter', function (evt) {
                //count = 0; //カウントリセット
                Timer6 = setInterval(
                    function(){
                        count6++; //カウントアップ
                        console.log(count6);
                        document.getElementById('like6').setAttribute('value', count6);
                    },
                    1000 //1000ms(1s)ごとに実行
                    );
                });
            this.el.addEventListener('mouseleave', function (evt) {
                clearInterval( Timer6 );
            });
        }
    });
</script>
```

_form.html.erb

```
<%= form_with(model: @room, local: true) do |f| %>
    <div class="form-group">
        <%= f.label :name, class: "control-label" %>
        <%= f.text_field :name,class: "form-control" %>
    </div>
    <div class="form-group">
        <%= f.label :description, class: "control-label" %>
        <%= f.text_area :description,class: "form-control" %>
    </div>

    <div id="works">
        <%= f.fields_for :works do |work| %>
            <%= render 'work_fields', f: work %>
        <% end %>
    </div>
    <%= f.submit 'Submit', class: "btn btn-primary" %>
<% end %>
```

_work_fields.html.erb

```
<div class='nested-fields card'>
    <div class="card-header">作品</div>
    <div class="card-body">
        <%= f.hidden_field :_destroy %>
        <div class="form-group">
            <%= f.label :name, class: "control-label" %>
            <%= f.text_field :name, class: "form-control" %>
        </div>
        <div class="form-group">
            <%= f.label :description, class: "control-label" %>
            <%= f.text_area :description, class: "form-control" %>
        </div>
        <div class="form-group">
            <%= f.label :image, class: "control-label" %>
            <%= f.file_field :image %>
        </div>
    </div>
</div>
```

_aframe_footer.html.erb

```
<div id="aframe-footer" class="container">
    <div class="flex-container">
        <div class="flex-item">
            <h1 id="name"> <%= room.name %></h1>
        </div>
        <div class="flex-item">
            <button id="show" class="btn btn-primary">説明を見る</button>
        </div>
    </div>
    <div id="desc-container">
        <p id="description"> <%= room.description %></p>
    </div>
</div>
```

resizeVR.js

```
const setHeight = () => {
    let elem = document.getElementById("aframe");
    //ウィンドウ縦幅を取得
    let wh = window.innerHeight;
    //ウィンドウ縦幅の85%を高さに指定
    elem.style.height = ( wh * 0.85 ) + "px";
}

//リロード・リサイズ時に実行
window.addEventListener("load", setHeight);
window.addEventListener("resize", setHeight);
```

toggleDescription.js

```
$('#show').on('click',function() {
    $('#desc-container').slideToggle();
});
```

aframe.css

```css
* {
    margin: 0;
    padding: 0;
    font-family: sans-serif;
}

img {
    object-fit: contain;
}

a-scene {
    z-index: 0;
    width: 100%;
}

#aframe-footer {
    z-index: 100;
    padding-top: 10px;
    padding-bottom: 10px;
}

.flex-container {
    display: flex;
}

.flex-item {
    margin-left: 30px;
    margin-right: 30px;
}

#desc-container {
    display: none;
    margin-left: 30px;
    margin-right: 30px;
}
```

form.css

```css
body {
    margin: 10px;
}

.nested-fields {
    margin-top: 5px;
    margin-bottom: 5px;
}
```

index.css

```css
#desc-container {
    margin-bottom: 10px;
}

.card {
    margin-bottom: 20px;
}
```