コンピュータ理工学特別研究報告書

題目

立体 360° 画像の表示とナビゲーション に関する研究

学生証番号　344360

氏名　笠原 凱

提出日　平成 29 年 1 月 31 日

指導教員　蚊野　浩

京都産業大学
コンピュータ理工学部

要約

撮影者を中心とする全ての風景を，ワンショットで撮影することができる 360° カメラの利用が広がっている．従来の写真が，画角によって切り取られた狭い範囲を見るものであるのに対し，360° カメラが写す 360° 画像は，記録した風景を仮想的に体験することが可能である．360° 画像が利用されているのは，不動産・観光業の Web サイトである．これらのサイトでは，360° の広視野空間を見渡すことができるビューワーを利用して，物件の内見などをすることができる．

本研究では，360° 画像を用いた立体 VR システムと 360° VR コンテンツを作成した．これらの開発に Unity5.5 を使い，360° 画像を観察する装置にヘッドマウントディスプレイである Oculus Rift を使用した．

立体 VR システムは，両眼視差を利用した立体 360° 画像を閲覧するものである．あらかじめ準備した，蚊野教授室の左目用 360° 画像と右目用 360° 画像を，2 つの球面の内側にテクスチャマッピングし，それらを Oculus Rift の左目ディスプレイと右目ディスプレイに表示した．その結果，蚊野教授室の室内を，臨場感のある立体的な 360° 画像として体験することが可能になった．

360° VR コンテンツは，通常の 360° ビューワーの画面内で，仮想キャラクタがナビゲーションを行うようにしたものである．ナビゲーションビューの仮想キャラクタにオープンソース系アイドルのユニティちゃんを使用した．ユニティちゃんの機能に歩行やジャンプといったアニメーション機能があり，動作をプログラムすることでアニメーションの操作を可能にした．蚊野教授室の室内を案内するように開発したナビゲーションビューは，概ね，期待どおりに動作した．しかし，ナビゲーションをするユニティちゃんの行動アクションにいくつかの課題が残った．

目次

1章 序論

360°カメラと専用アプリを使用することで，全周 360°の範囲を１枚に収めた 360°画像や 360°動画を簡単に撮影することができる．360°画像や動画を 360°ビューワーで見ると，撮影地点を中心として上下左右 360°の空間を見渡すことができるため，まるでその場にいるかのような臨場感を味わうことが可能である．360°画像を使用したビューワーの代表例として Google ストリートビューがある．

360°画像を楽しむために，ヘッドマウントディスプレイ(HMD)や，モバイル端末を装着した VR ゴーグルを用いることが一般的である．これらの装置では，HMD やモバイル端末に組み込まれたセンサが頭部の動きを検出し，それに応じて視聴画像を変更する．そのため，マウスによる画面移動やモバイル端末画面をスライド操作することに比べ，画像が表現する３次元空間の中を見回している感覚になる．本研究ではこれを発展させ，左目画像・右目画像からなる立体 360°画像を HMD で観察する立体 VR システムを開発する．このシステムでは，視差を利用して画像内の物体を立体表示することで，より臨場感の高い VR 体験を実現することが目標である．

不動産・観光業などのビジネスでは，360°ビューを用いて，賃貸物件や観光地の情報を提供している．多くの 360°ビューは，画像内の物件を説明するために，ユーザの操作に応じてテキストや画像を表示する．インタラクティブな情報提供が存在する位置は 360°画像上にマーカーなどで明示するが，その情報の重要度や種類がユーザに不明なことが多い．本研究では，ナビゲーション機能を実装した 360°ビューを開発することで，ユーザに対する情報提供の効率性および快適性を追求する．なお，本論文では，ユーザの操作に応じて表示される情報を「インタラクティブ」とよぶ場合がある．

以下，２章では開発環境および開発に使用された技術について述べる．３章では立体 360°ビューの開発について述べる．４章では３章で開発したコンテンツを基盤にした 360°ナビゲーションビューの開発について述べる．最後に５章で結論を述べる．

2 章　HMD を使った 360°ビューの開発環境

２章では 360°画像の生成方法や，本研究で開発するシステムの開発環境などを説明する．

2.1　360°画像の撮影と表示

標準レンズで撮影した写真画像は，撮影者の前方を中心とした 45°程度の範囲を写している．これに対して 360°カメラは，カメラを中心とした全ての方向の視野をカバーすることができる．360°カメラで撮影した画像を 360°画像とよぶ．

360°画像の全体を一枚の画像として提示する場合，Equirectangular 画像（正距円筒図法で球面を画像化したもの）として表示することが多い（図 1 上の画像）．360°画像の閲覧方法として，図 1 に示すように，視線方向を指示し，その方向を中心とする矩形画像を表示することが一般的である．ユーザが指示する視線方向に応じて表示画像を変えると，ユーザにある程度の臨場感を与えることができる．

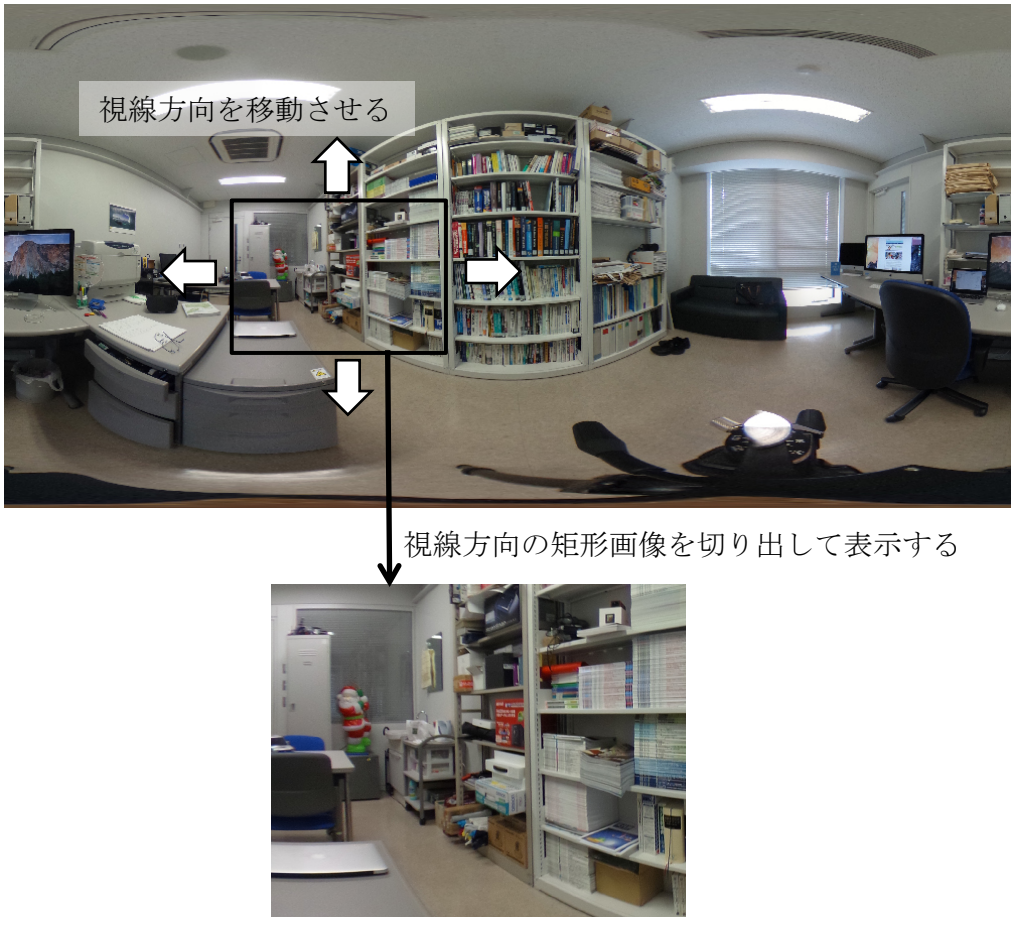


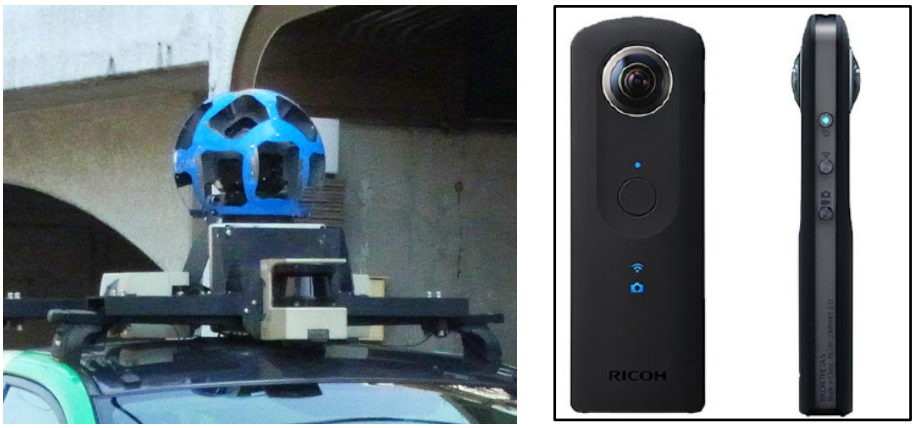
図 1 360°画像と表示

2.2 360°画像の利用

360°画像は，1995 年頃に開発されたアップル社の QuickTime VR で利用された．当時は，全周パノラマ画像などと呼ばれ，くるくる回る画像として注目された．

2007 年に Google ストリートビューのサービスが開始された．これは，自動車の屋根に搭載した全天球カメラで，車を走らせながら大量の 360°画像を撮影する．そして，Google マップと組み合わせて 360°画像を閲覧することで，仮想的に現地を移動し見回すことを可能にする．ストートビューで利用された全天球カメラ（図 2 左）は，複数のカメラを球体状に組み合わせたものである．一般人が簡単に利用できるものではないため，360°画像の利用の広がりも限られていた．

なお，360°カメラ，360°画像には，全天球カメラ，全周パノラマ画像などの名称もあるが，この論文では，基本的に 360°カメラ，360°画像とよぶことにする．



https://commons.wikimedia.org/wiki/File:Google_Street_View_Car_in_Tokyo.JPGの一部
RICHOのWebサイトの画像の一部

図 2 Google ストリートビューに用いた全天球カメラと RICHO THETA S

RICOH THETA S（図 2 右）のような 360°カメラが，360°画像の利用を広めた．不動産物件や名所・ホテルの紹介などに利用されている．360°画像のビジネス利用も活発になっている．Smart360[2]はそのような Web サイトの例である．そこでは，360°画像を使用した空間上に，文字・写真・音声・動画などの情報を付けた 360°ビューを提供している．

2.3　本研究における開発環境

本研究では 360°画像を使ったアプリケーションの開発に Unity を使った．Unity はマルチプラットフォームに対応したゲームエンジンである．Unity で作成したコンテンツは Android・iOS などの主要なモバイルデバイス，VR デバイス，パソコン，コンソールゲーム機，TV プラットフォームや Web 向けに出力が可能である．本研究で作成

した立体360°ビューと360°ナビゲーションビューは，HMDであるOculus Rift(図３)をVRデバイスとして利用する．

Unityで利用するプログラミング言語はC#・JavaScript・Booの３つである．言語使用比率はC#が約80%，JavaScriptが約20%の割合を占めており，Booの使用率は０%に近い値である．JavaScriptはUnity独自の言語仕様となっており，本来のJavaScriptとは違う部分がある．UnityではC#でしか利用できない機能もあるため，本研究ではアプリ開発の言語としてC#を採用した．

図3 Oculus Rift

2.4 Oculus Rift

Oculus Rift（図３）は2016年3月から販売されているHMDである．広視野で，頭の動きに表示を追随させるためのヘッドトラッキング機能を持っている．主な仕様を表1に示す．

表1 Oculus Riftの主な仕様

項目	仕様
表示パネル	有機EL，1080×1200画素×2枚
リフレッシュレート	90Hz
視野角	水平視野角:110°
ヘッドトラッキング	赤外LEDとカメラを使う方式

3 章　立体 360° ビューの開発

立体 360° 画像を Oculus Rift で閲覧することができる立体 360° ビューを開発した.

3.1　立体 360° 画像の生成

本研究で扱う立体画像は両眼視差に基づくものである．人間の左右の眼球の間隔は大人で 60mm 前後，子供では 50mm 前後である．このため，同じ世界を見ても左目と右目には異なった像が映る．そのズレを“両眼視差”とよぶ．２枚の画像に存在する両眼視差を脳が処理することで遠近を判別し，立体視が可能になる．

立体 360° 画像は，左目 360° 画像と右目 360° 画像をからなる一組の 360° 画像である．左右の 360° 画像の間には，あらゆる視線方向で，正しく視差が再現されている必要がある．本研究では，今年度の特別研究 II[5]で生成した図４の２枚の 360° 画像を利用した．２枚の画像を比較すると，左目 360° 画像がやや右に寄っており，視差が有ることがわかる．視差は，近い被写体で大きく，遠い被写体で小さい.

左目360° 画像

右目360° 画像

図 4 立体 360° 画像の例

3.2 Unity と Oculus Rift を使った立体 360° ビュー

Unity はゲームアプリを開発する統合開発環境である．一つのゲームをプロジェクトという単位で管理する．Unity による開発では，シーン上にオブジェクトを配置し，動作を付与する．シーンは関係のあるオブジェクトのまとまりである．オブジェクトには，球体や木のように形を持つものと，照明やカメラ，オーディオのように，プレイ画面では具体的な形が表示されないものがある．あらかじめ定義されている基本的なオブジェクトの機能は単純である．これに，コンポーネントを追加することでオブジェクトに機能や動作を加えることができる．

ここからは Unity を使用した立体 360° ビューの開発方法について説明する．このプロジェクトの作成手順を簡単に説明すると以下のようになる．

（1）２つの球体オブジェクトを配置する．

（2）それぞれの球体オブジェクトの内部にカメラを配置する．

（3）球体オブジェクトの内側に 360° 画像をテクスチャマッピングする．

（4）Oculus Rift のヘッドトラキングにあわせてカメラの視線を移動させる．

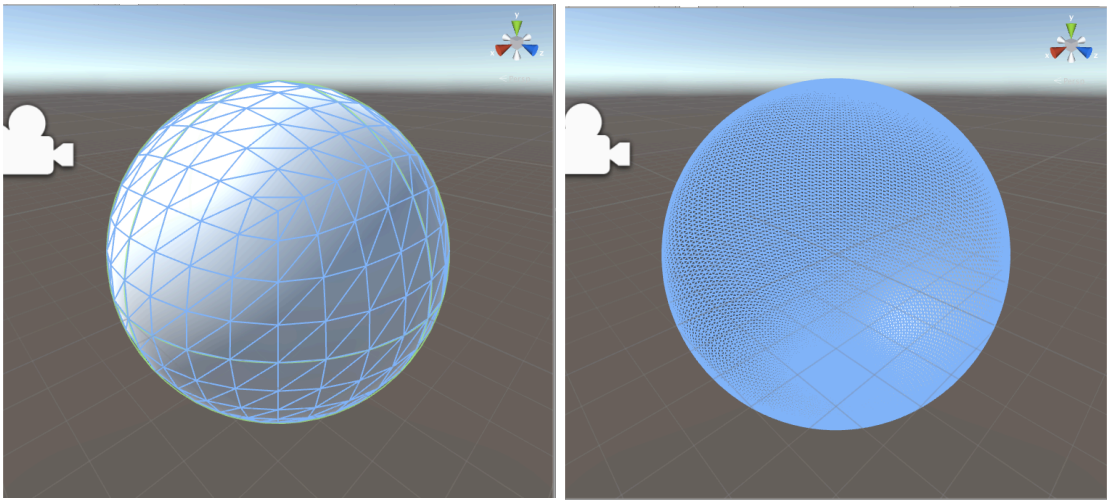
これらについて，説明する．

（1）２つの球体のオブジェクトを配置する

360° 画像を表示する方法として，2.1 節では，Equirectangular な 360° 画像から視線方向の矩形領域を切り取り，それをディスプレイに表示すると説明した（図 1）．厳密に言えばこれは正しくない．Equirectangular 画像はシーン中の直線が湾曲する性質があるので，これを補正しなければならない．360° 画像を球体の内側表面にテクスチャマッピングし，それを球体の中心においたカメラに投影することで正しい表示画像になる．

これを行うために Unity のシーン中に球体オブジェクトを配置した．本研究では，高精細天球モデルの Sphere100[1]を使用した．これは，Unity にデフォルトで定義されている Sphere に比べて高精細である．その比較を図 5 に示す．360° 画像を Sphere にテクスチャマッピングした場合，ポリゴンのエッジが知覚されたが，Sphere100 では滑らかに見えた．

立体 360° 画像を表示するために，２つの球体オブジェクトを配置した．２つの球体の位置関係に制約はないが，重ならないようにする必要はある．



(a) Sphereのポリゴン　　(b) Sphere100のポリゴン

図 5 球体のポリゴンの比較

（2）球体オブジェクトの内部にカメラを配置する

Sphere100 の中心にカメラを配置する（図 6 上）．この状態でカメラに映るゲームビューを図 6 下に示す．球体の内部が見えていることがわかる．

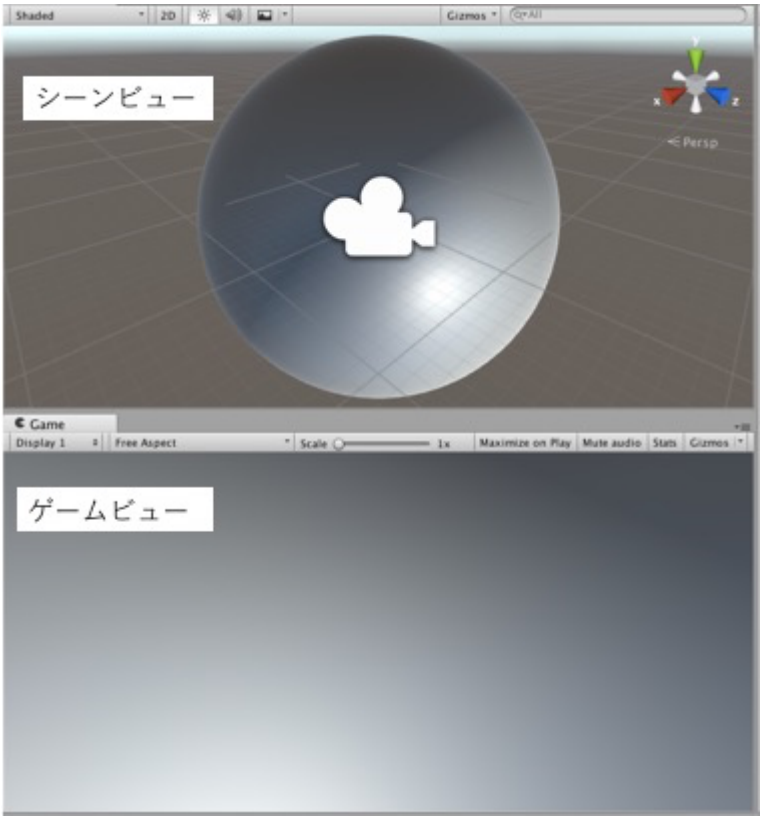


図 6 Sphere100 を配置したシーンビューとカメラに映るゲームビュー

（３）球体オブジェクトの内側に 360° 画像をテクスチャマッピングする

図７の球体オブジェクトに図１の画像をテクスチャマッピングした時の，シーンビューとゲームビューを図７に示す．貼り付けた画像の Shader 設定を Unlit/Texture にする.Unlit/Texture は Shader にデフォルトで存在するオプションの一つである．Shader は，３次元 CG において，シェーディング（陰影処理）を行うプログラムである．今回使用した Unlit/Texture は，照明による影響をなくす設定である．これを設定しない場合，照明と球面の位置関係により，テクスチャ画像に濃淡変化があらわれる．

図 7 Unlit/Texture を設定した場合

左右２つの球体オブジェクトにテクスチャをマッピングし，完成させたシーンを図８に示す．

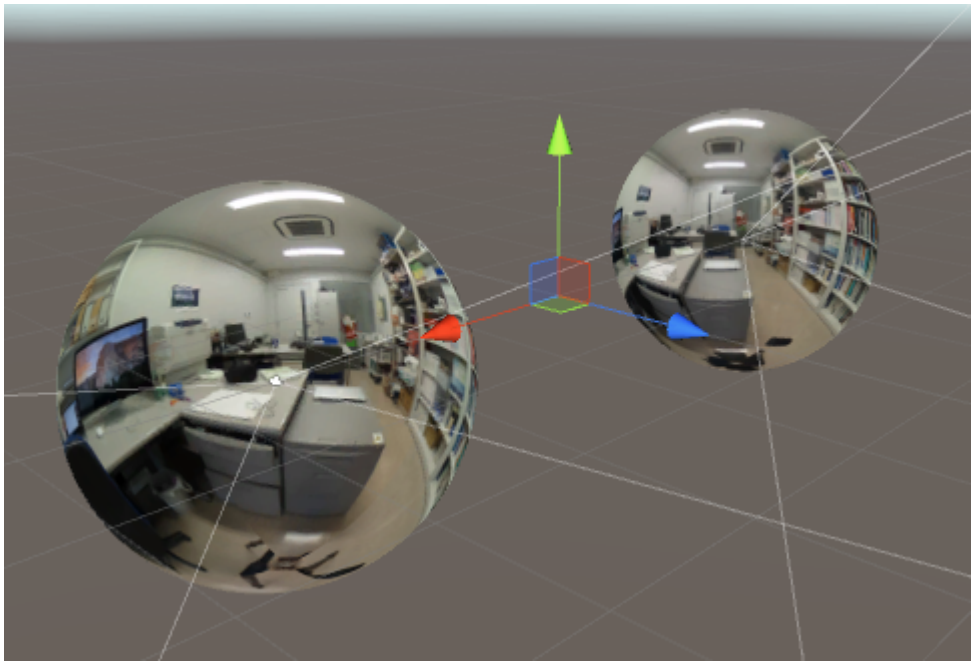


図 8　２つの球体オブジェクトに 360°画像をマッピングして完成させたシーン

（４）Oculus Rift のヘッドトラッキングに合わせてカメラの視線を移動させる

Unity を Oculus Rift に対応させると，ヘッドトラキングの結果が自動的にカメラの動きに変換される．また，カメラの画角も，自動的に Oculus Rift の視野角に合う．Unity を Oculus Rift に対応させる方法は，File メニューから Build Settings を選ぶ．Build Settings のウインドウで，Player Settings...をチェックすると PlayerSettings のパネルがあらわれる．その中にある Virtual Reality Supported のチェックボックスをチェックする．

図 4 の蚊野教授室の立体 360°画像を用いて，立体 360°ビューの動作を検証した．その結果，通常の 360°画像と比較して両眼視による立体感を得ることができ，臨場感が増すことを確認した．また，次のような課題があった．

（１） 表示画像の解像度（画素数）が不十分である．

（２） 頭部が平行移動した場合の視点移動を再現することができないので，違和感が生じる．

4 章　360°ナビゲーションビューの開発

3 章で作成した球体オブジェクトを改良し，Smart360 を参考にした 360°画像コンテンツを製作した．これは，素材として蚊野教授室の 360°画像を利用し，室内に配置した学生の研究テーマを紹介するものである．ビュー内の一部の対象物にはインタラクティブを設定した．その一例として，Unity の GUI を利用してテキスト表示したものを図 9 に示す．GUI は画像や図形アイコンを利用し，ユーザに情報の提示や操作の受付をする方法の 1 つである．テキストを表示するには画像内の対象物(図 9 の場合には，Oculus Rift とラジコンカーが写っている領域)を左クリックすることで表示できるようプログラムされている．

なお，３章では球体オブジェクトを２つ使用して立体視を可能にしていたが，４章では球体オブジェクトを１つだけを使用し，立体視は実装していない．

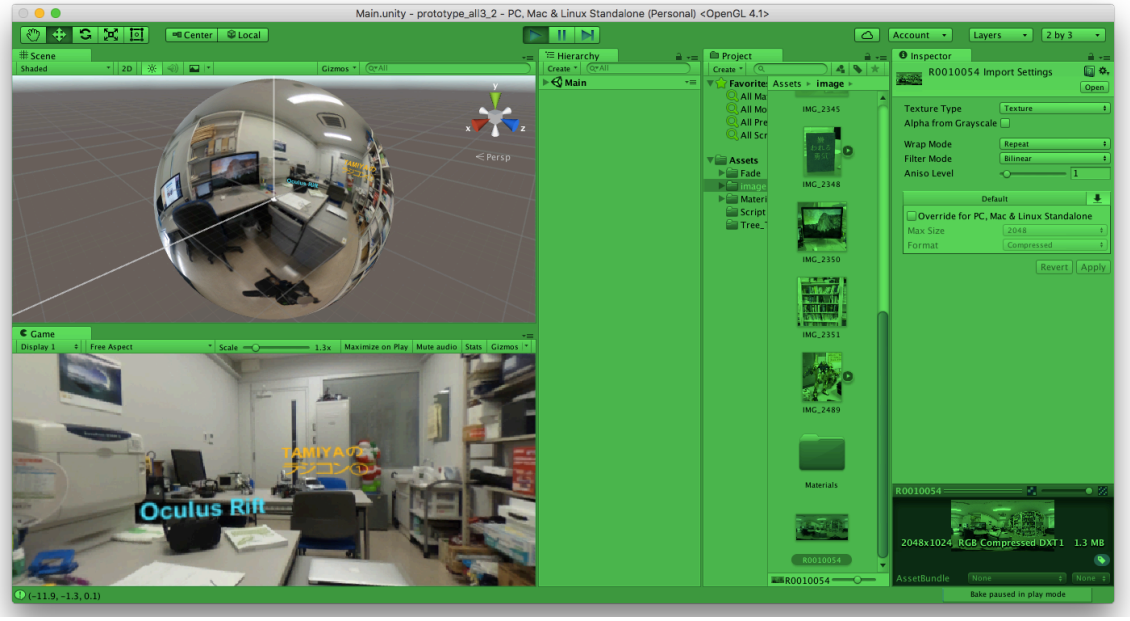


図 9 教授室の 360°ビューとテキストによるインタラクティブの表示例

4.1　説明用の文章や画像の表示

Smart360 の 360°ビューに，ビュー上に表示されたポイントをマウスクリックすることで対象物の画像などの情報が表示されるインタラクティブがある．このようなインタラクティブは 360°ビュー上に画像が拡大表示される場合が多く，ポイントをクリックするとすぐに画像が表示される(図 10)．このようなインタラクティブを実装した．しかし，360°ビューをパソコンで観察するのと異なり，Oculus Rift で観察すると，クリック後に突然画面が切り替わるのは違和感があるのではないかと考えた．そこで，表示画面が変更する際にトランジションを加えることで，違和感を緩和した．

トランジションは，テレビや映画で場面転換を行う際の，フェードアウトなどの効果のことである．

図 10 画像が拡大表示されるインタラクティブの例

今回は FadeCanvas[3]というトランジションのプログラムを改良し，プログラムに組み込んだ．トランジションを発生させたのは，元の 360° ビューである Main のシーン(図 9)と，説明用の画像および文章を表示するシーン（image_sence）(図 11)の間である．トランジション中の画面の例を図 12 に示す．図 12 は Main シーンの中心から渦巻き状態のトランジションが画面を覆い隠し，その後，image_sence へと画面が移り変わる時のものである．トランジションは Main シーンから image_sence へ移動する場合と、image_sence から Main シーンへ移動する場合のいずれでも発生する．

図 10 における拡大表示された画像の下部にある「本格バリ雑貨を…」のように，画像と共に説明の文章が添付されていることがある．この場合，画像と共に表示するため文字数は少なく，文字が小さくなる．そこで，image_sence で表示する説明文のテキストをスクロール可能にすることで、文字数の制限や文字の小ささをカバーした．スクロールしている状態を図 13 に示す．

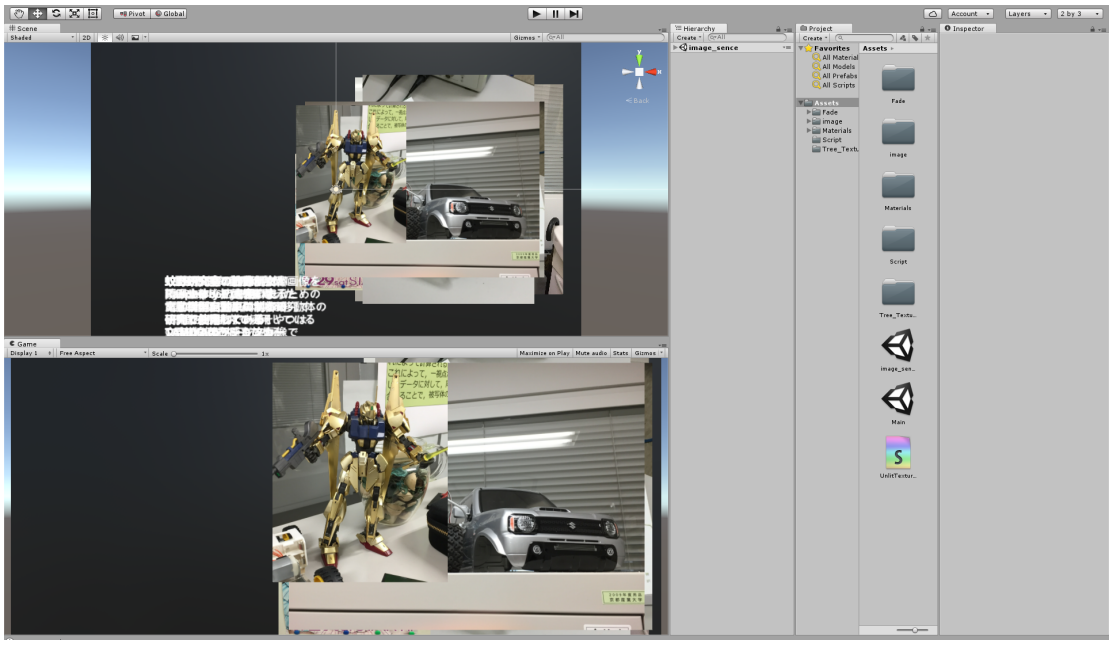


図 11 説明用シーン

図 12 トランジション中の画面

図 13 説明用文章のスクロール

4.2　仮想キャラクタを使ったナビゲーション

Web ブラウザで利用する 360° ビューにインタラクティブな表示を加える場合，多くのサイトではインタラクティブが存在する場所に目印マーカーを表示している．しかし，そのインタラクティブが画像を表示するものか，テキストを表示するものか，関連サイトに移行するものか，といった情報が不明である場合が多い．このような設計ではインタラクティブの操作はユーザに任されてしまい，製作者側が知って欲しい情報が明確になりにくい．これを改善するために，ビュー内に仮想キャラクタを配置して，ナビゲーションを行うビューを開発した．これにより，インタラクティブが存在する場所や，インタラクティブが提供する情報の種類，インタラクティブに対して起こせるアクションなどを容易に理解できる．また，製作者の意図を仮想キャラクタに託すことも可能である。

仮想キャラクタとして，Unity Technologies Japan が開発したオープンソース系アイドルのユニティちゃん[3]の SD 版（図 14)を使用した．ユニティちゃんにはデフォルトの行動アクションが多数あり，プログラムすることで行動を操作することが可能である．

ユニティちゃんのナビゲーションプログラムによる動作を以下に記述する．

（1）ナビゲーションビュー内をマウスで右クリックする．

（2）クリックした位置の付近にインタラクティブな情報をもつ対象物が存在するならば（3）に行く．そうでなければ（1）に戻る．

（3）ユニティちゃんの視線および体躯の向きをクリックした位置に向ける．

（4）クリックした位置にユニティちゃんを移動させる．その際に歩行アクションを実行する．

（5）移動後ユニティちゃんの視線および体躯の向きを撮影カメラに向け，歩行アクションを停止させる．

（6）ユニティちゃんがコメントによるナビゲーションを行う．

（6－A）対象物に対するテキストが表示できる場合

「対象物を左クリックすることでコメントが表示できます」と表示する．

（6－B）シーンを変更して説明画面を表示する場合

「対象物を左クリックすることで説明用画面に移行できます」と表示する．

上記のプログラムを繰り返す．クリックした位置の付近にインタラクティブがない場合ではユニティちゃんは反応を示さず，カメラ目線で直立不動の状態をとる．

図 14 SD ユニティちゃん

4.3　考察

３章で作成したプロジェクトを改良し，従来の 360°ビューとは異なる新しいコンテンツを開発した．課題として次のことがあげられる

（1）立体 360°画像に対応していない．

（2）ユーザがクリックした位置にユニティちゃんの目線と姿勢が向くが，その位置に到達するまで目線を離さない．そのため，ユニティちゃんが位置に近づくほど，徐々に顔が下がっていくため不自然である．

（3）クリックした位置に到達すると目線と体躯を唐突にカメラに向けるため，動作が早く不自然である．

（4）ユニティちゃんが行動アクションを起こすにはインタラクティブ付近の位置をクリックする必要あり，クリックがない限り行動しない．

5章　結論

本研究では Unity と Oculus Rift を使い，立体 360° 画像を VR 体験できるシステムを開発した．このシステムは，あらかじめ準備した左目用 360° 画像，右目用 360° 画像を，Unity で定義した２個の球面にテクスチャマッピングし，視線方向の画像を頭部の動きに応じて HMD に提示するものである．蚊野教授室の室内を撮影した立体 360° 画像を用いて動作検証し，臨場感の高い VR 体験が可能であることを確認した．

次に，360° ビューに，説明用画像やテキストをインタラクティブに表示する機能を加えた．さらに，仮想キャラクタによるナビゲーション機能を付加した VR コンテンツを作成した．仮想キャラクタは，360° ビューの中である程度，動作可能になったが，以下のことが課題として残った．

（１）立体 VR システムで動作させることができていない．

（２）移動中，ユニティちゃんの頭が下がった状態になる．

（３）ユニティちゃんの急なアクションに違和感がある．

（４）ユニティちゃんがマウスでクリックがないと行動しない．

参考文献

[1] 高精細天球モデル Spherer100
https://t.co/Wd6Osuqm6X
[2] smart360
http://www.agentec.jp/panorama.html
[3] FadeCanvas
http://tsubakit1.hateblo.jp/entry/2015/11/04/015355
[4] ユニティちゃん
http://unity-chan.com/contents/guideline/
[5] 高木 亮佑,「4枚の 360° 画像のつなぎ合わせによる立体 360° 画像の生成に関する研究」, 京都産業大学 コンピュータ理工学部 特別研究 II, 2017 年 1 月.

謝辞

本論文を作成にあたり,丁寧な御指導を賜りました蚊野浩教授に感謝いたします.

付録　本研究で開発したプログラムの説明

表２　本研究で作成したプログラム一覧

名前	説明
MouseDrag.cs	360°ビュー内でマウス操作（右クリック・ドラッグ）を行った際，カメラの視点を変更する．Unity を Oculus Rift に対応させた場合には，ヘッドトラキングの結果が自動的にカメラの動きに変換される．
MouseEvent.cs	360°ビューでマウス操作（左クリック・右クリック）を行った際の座標点を取得する．座標点からテキスト表示・非表示の判別を行う．
ObjectEvent.cs	スクリプト MouseEvent.cs で判別が行われたテキストに対して表示または非表示を実行する．また，image_sence シーンに移動する場合はトランジションを発生させ，シーンを移動する．
ImageCameraEvent.cs	360°ビューでマウス操作（右クリック）を行った際にトランジションを発生させ、Main シーンに移動する．
ImageEvent.cs	Image_sence シーンに移動した際に，表示する説明文および画像の判別を行い、表示する．説明文を表示後，スクロールを行う．
UnityMove.cs	Unity ちゃんの移動・行動アクション・テキスト表示を行う

MouseEvent.cs，ObjectEvent.cs は Main シーンの MainCamera に，MouseDrag.cs は Sphere100 にコンポーネントされている Script である．ImageCameraEvent.cs，ImageEvent.cs は image_sence シーンの ImageCamera にコンポーネントされている Script である．