令和元年度　東三河地域防災協議会受託研究　最終報告書

拡張現実を用いた浸水・暴風雨・土砂災害疑似体験アプリの開発と防災教育における活用

令和３年３月

研究代表者　宇野 新太郎

愛知工科大学工学部

情報メディア学科 教授

研究分担者　板宮 朋基

愛知工科大学工学部 研究員

神奈川歯科大学歯学部 教授

目次

概要

2018 年 7 月の西日本豪雨や 2019 年 10 月の台風 19 号などにおける教訓から，災害発生時において迅速かつ的確な避難を可能にするための対策が求められている．適切な避難行動を支援するために各自治体ではハザードマップの整備が進められているが，地図に表示された情報を読み取り脳内で正確にその光景をイメージできる人は多くない．そこで本研究では，平時における風水害への危機意識の向上を目的として，Apple iPhone 11 と Android スマートフォンを用いて暴風雨時と土砂崩れにおける災害発生状況を現実風景に重ねて CG（コンピュータ・グラフィックス）表示し，疑似体験できる拡張現実(AR)スマートフォンアプリを開発した．一般的に普及しているスマートフォンを用いるため汎用性が高く複数人同時での体験が可能であり，避難訓練において幅広く活用できる．拡張現実アプリ開発環境 Apple ARKit3 や Google ARCore DepthAPI の機能を活用し，人物や家具などの形状を自動認識し，暴風による飛来物の落下状況や豪雨の降り方や浸水発生および土砂崩れの状況を，一般的に普及しているスマートフォンにおいてリアルに表現することが可能になった．そのため，本アプリの体験者は風水害による災害のリスクを「自分のこと」として実感できる．画面操作によって飛来物の速度や降雨の向きおよび水位・流速や漂流物の有無や土砂崩れの角度を設定することが可能である．評価の結果，本システムは危機意識の向上に有用であることが示された．

１．はじめに

2015 年 9 月の関東・東北豪雨や 2018 年 7 月の西日本豪雨，さらに 2019 年 10 月の台風 19 号など日本各地で暴風雨によって大規模な風水害が発生している．2019 年の台風 19 号がもたらした人的被害は，死者 86 人，行方不明者 3 人，重軽症者 476 人に及び, 建築物被害は全国で 8 万棟以上の住宅に被害が確認された[1]. 台風 19 号において，最大瞬間風速がおよそ 25m/s 以上で倒木・家屋損壊・大きな飛来物・停電という被害が目立ち始めた[2]．2017 年の内閣府による「防災に関する世論調査」では，「防災訓練が行われていることは知っていたが参加や見学をしたことがない」と答えた人が 30. 7%，「訓練が行われていることを知らなかった」と答えた人が 24．0%となっている[3]．防災に関する意識調査のグラフを図 1 に，被害が目立ち始める風速を図 2 に示す．

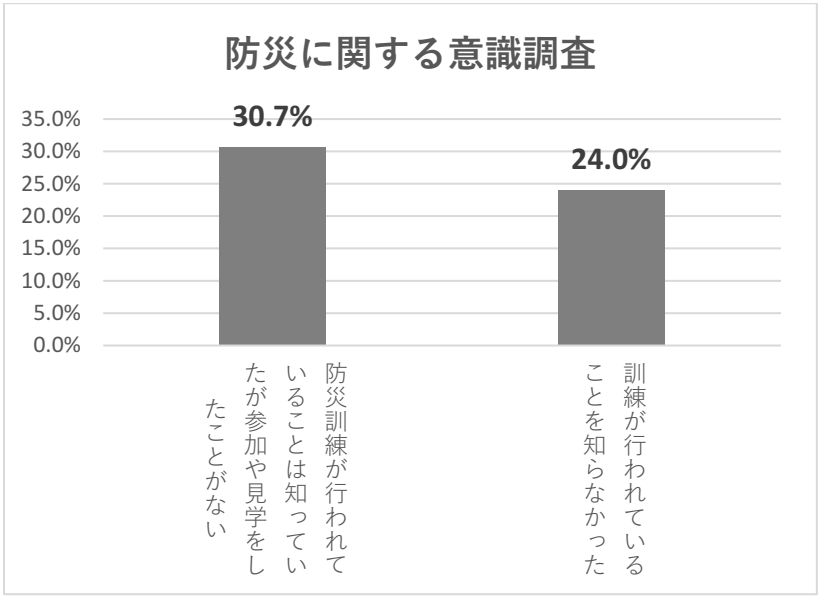

図 1　防災に関する意識調査

倒木	最大瞬間風速 24m/s
大きな飛来物	最大瞬間風速 26m/s
家屋破損	最大瞬間風速 27m/s
停電	最大瞬間風速 26m/s

図２　被害が目立ち始める風速

以上のことから，災害発生時における迅速かつ安全な避難を可能にするための対策が求められている．これまでに，適切な避難行動を支援するため各自治体でハザードマップの整備が進められているが，地図に重畳された情報を読み取り頭の中に正確にイメージできる人は多くない．そこで実際の行動に結び付けるためのより効果的な伝達手法が求められている[4].

本研究では，暴風雨による風水害に対する平時からの危機意識の向上を目的として，一般的に普及していて入手が比較的容易なスマートフォンと紙製簡易ゴーグルを用いて，現在位置における台風など暴風雨による風水害や土砂崩れが発生した状況を現実風景に重ねてリアルに表示し，没入体験できる AR（拡張現実）アプリを開発した．Apple による拡張現実アプリ開発環境 ARKit3 の People Occlusion では，人物の輪郭を自動認識し，現実空間の人物の背後には CG が表示されない．Google ARCore DepthAPI は，スマートフォンのカメラ映像から空間を認識し，壁や家具などの形状を自動認識し，現実空間の物体の背後には CG が表示されない．そのため，暴風雨による飛来物や浸水や土砂崩れの危険性をよりリアルに表現することが可能になった．降雨表現や風速，水位や流速，土砂崩れの角度を任意に変化させることも可能である．

拡張現実(AR: Augmented-Reality)とは，現実に見えている風景に，コンピュータ上で処理された情報を重ね合わせ，情報を付加する技術であり近年様々な分野で活用が期待されている[5] [6]．また，現在多くの人がスマートフォンを利用している．平成 26 年(2014 年)調査時，全年代で 6 割以上，20 代では 9 割以上の人がスマートフォンを利用している[7]．平成 26 年(2014 年)モバイル機器等の利用率(全年代・年代別)のグラフを図 3 に示す．加えて，パソコン並みの性能を持つスマートフォンも年々多くなり，今までは処理が重く実現が難しかったことが実現可能になっている．

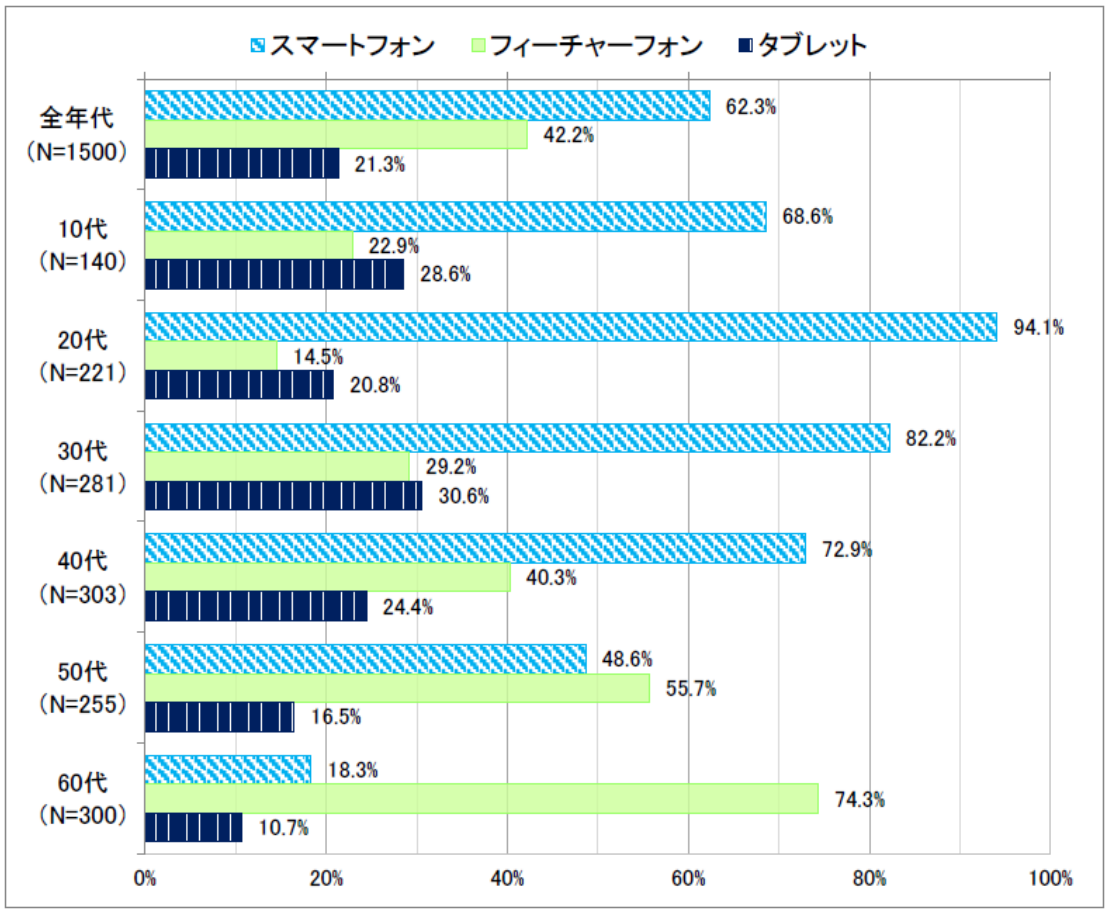

図 3　平成 26 年(2014 年)モバイル機器等の利用率(全年代・年代別)

本システムでは，一般的なスマートフォンと紙製簡易ゴーグルを用いるため，運搬が容易である．特殊な機器は必要なく，汎用性が高い．紙製ゴーグルは 1 個当たり 1000 円程度で調達可能であるため，従来のシステムに比べて低コストで実現可能であり，避難訓練等の防災イベントで複数台同時に利用することができる．

2．先行研究

2．1　3D-VR(3 次元バーチャルリアリティ)を活用した災害状況の再現・対応能力訓練システム

香川大学危機管理研究センター[8]は，災害時の状況を再現できる 3D-VR(3 次元バーチャルリアリティ)を活用した対応能力訓練システムを開発した．想定を超える災害状況を再現し，訓練体験者がその危機的な状況で判断し，意思決定を行い，行動を起こすという一連の訓練を行う．システムにはカメラやセンサが設置されており，訓練実施後に対応を振り返ることもできる．災害状況の再現・対応能力訓練システムの概要図を図 4 に示す．3D-VR（3 次元バーチャルリアリティ）を活用した災害状況の再現･対応能力訓練システムを利用して公開訓練をしている様子を図 5 に示す．

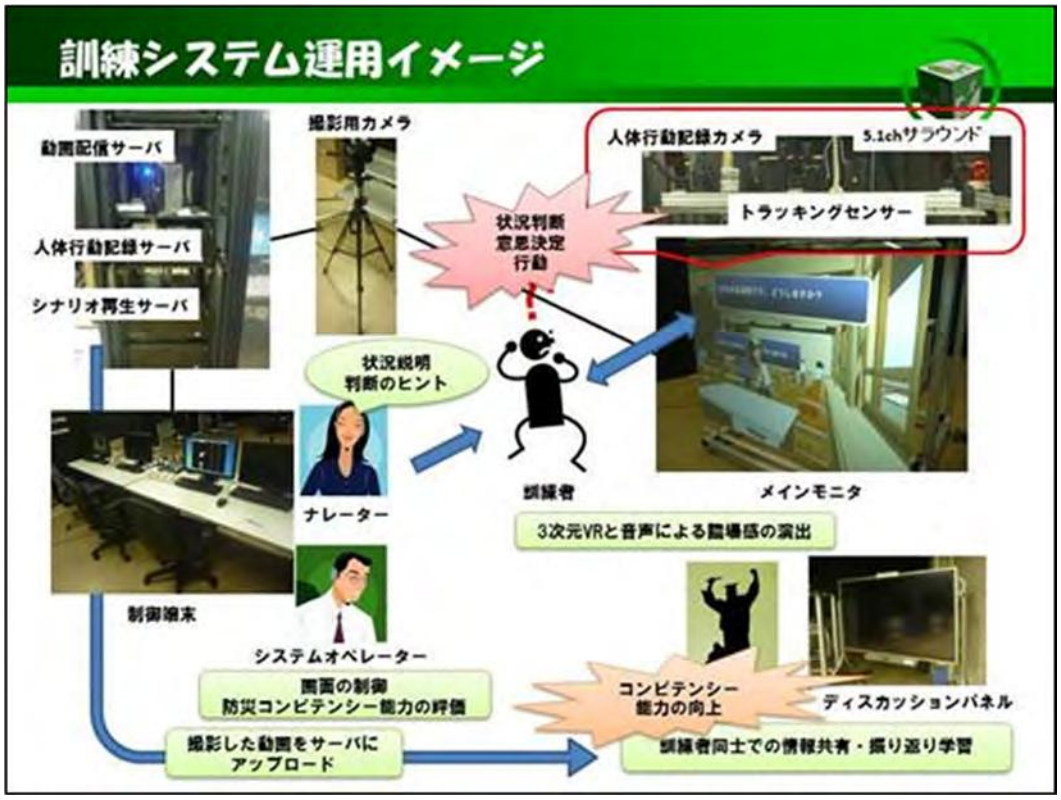

図 4 災害状況の再現・対応能力訓練システムの概要図

図 5 公開訓練の様子

2．2 VR 防災体験車

東京消防庁は，VR 防災体験車を製作し[9]，2018 年 4 月から運用されている．1 年間で約 5 万 8 千人が体験を行った．全長約 11．9 メートルの車内には，MX4D（アメリカ

合衆国の MediaMation 社によって開発された 4D 映画システムであり、座席の動き、臭い、水などで映画の演出を行う.）の椅子が備え付けられ，最大 8 人で同時に VR 映像を見ることができる．地震・火災・風水害の 3 つのストーリーに合わせて座席が動き、火災の熱や水しぶきなどの効果で，災害現場をバーチャル体感できる．
VR 防災体験車の概要図を図 6 に示す.

図 6　**VR** 防災体験車の概要図

2．3 防災 VR／暴風雨編

株式会社アイデアクラウド[10]では，台風の風や雨の被害を VR 空間上で体験可能な防災 VR 台風／暴雨風編を開発した. 作製した 3D マップ上で都市空間を表現している．一人称視点で表現されている．防災 VR／暴風雨編の表示例を図 7 に示す.

図 7 防災 VR／暴風雨編の表示例

2．4 AR 浸水状況疑似体験アプリ（2018 年度版）

愛知工科大学の板宮らは，赤外線を用いた 3D 奥行きセンサ(ToF センサ)が搭載されたスマートフォン ASUS 社製 Zenfone AR と，紙製ゴーグルを用いた AR 浸水状況疑似体験アプリを開発した．3D 奥行きセンサが地面を感知してスマートフォンの高さ位置を精密に把握し，任意の高さに水面が表示される[11]．5m 程度の周囲物体の形状も認識し，現実風景にある物体の背後には CG が表示されないオクルージョン（遮蔽）表現が可能であるため，リアルな浸水表現が可能である．浸水状況疑似体験アプリ(2018 年度版)の表示例を図 8 に示す．しかし，ASUS 社製 Zenfone AR は生産が終了しており，2020 年 1 月現在入手は困難である．また，拡張現実開発環境 Google Tango もサポートが終了しているため，今後の発展性と継続性に乏しい．

図 8 浸水状況疑似体験アプリ（2018 年度版）の表示例

2．5 本研究の位置づけ

本研究の位置づけを図 9 に示す．

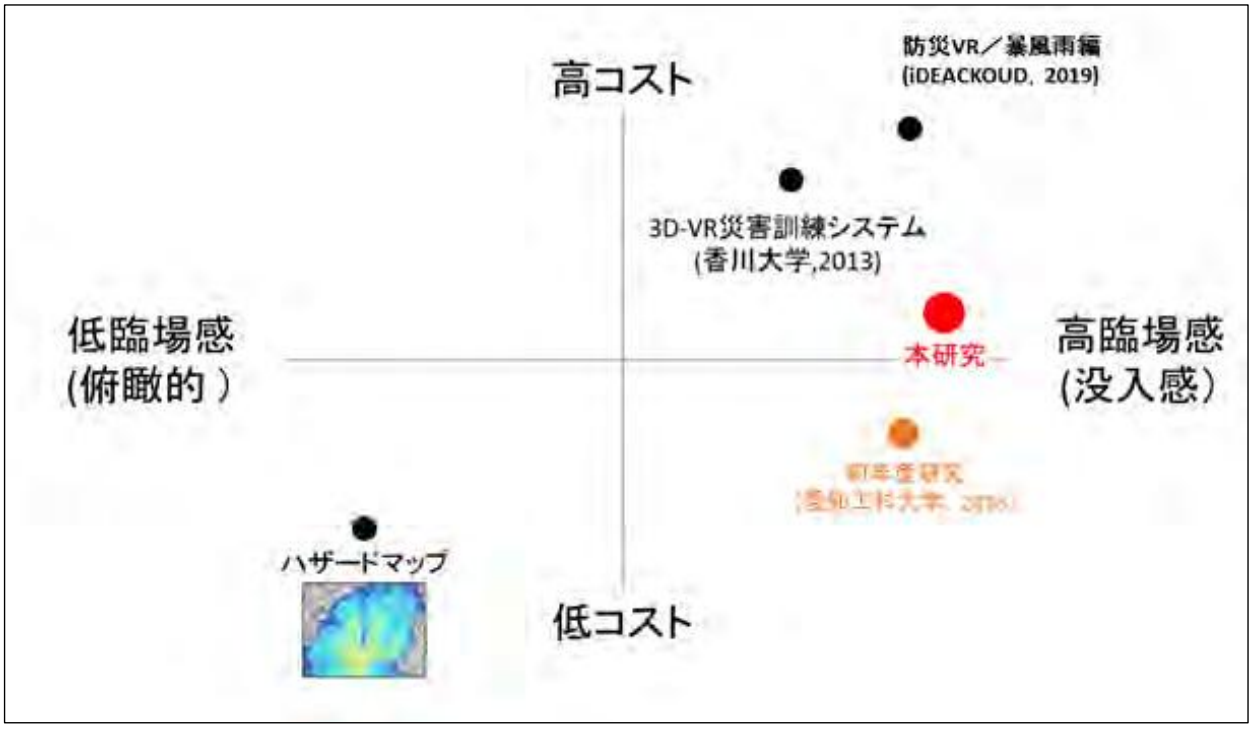

図 9 本研究の位置づけ

本研究は，拡張現実(AR)技術および iPhone 11 および Android スマートフォンと簡易ゴーグルを用いることで，現実空間に 3D-CG で作成された暴風雨や浸水および土砂崩れの発生状況をリアルに重ねて表示できる．本研究で開発したアプリはスマートフォンのみで体験可能である．そのため，特別な施設を備えた施設へ行く必要はなく，自宅や学校・職場など身近な環境で暴風雨による災害発生状況を疑似体験できる．VR は視野全体を 3D-CG または実写映像で表現するため，VR 災害疑似体験システムは体験場所毎の 3D-CG モデリング作業や撮影・編集作業が必須であり，初期導入費用は数十万円～数千万円と高額であり，体験場所の追加制作にも百万円単位の多額の費用が必要である．その点，AR は現実空間に 3D-CG を重ねて表示できるため，どの場所においても直ちに活用が可能であり，体験場所の追加に伴う費用は発生しない．AR の先行研究である愛知工科大学板宮研究室の 2018 年度研究では，スマートフォン ASUS 社製 Zenfone AR (Android7.0)を使用した浸水状況疑似体験アプリを開発した．
Zenfone AR に装備された 3D 奥行きセンサを利用して高さ位置を精密に検出することや周囲の物体の 3 次元形状を認識し，設定された水位以上に物体には水面がかからないようにリアルタイムにオクルージョン（遮蔽）処理を行う．オクルージョン処理はスマートフォン内のみで行われ，遅延なく表示される．壁などの構造物や家財，人物を回り込むように水面が表示され低い水位の場合でもリアルな浸水表現が可能になったため，より危険性を実感できる．また，実スケールの水位計を配置できる．画面上のボタンで水位を変更し画面上に現在の水位を表示できる．水中表現及び水流の変更も可能にした研究であった．しかし，Zenfone AR は生産終了のため継続的な調達が困難になった．アプリの開発に必須である拡張現実開発環境 Google Tango のサポートも 2018 年 3 月 1 日に終了したため，持続的発展は困難になった．そこで本研究では，一般的に普及していて入手が比較的容易な iPhone 11 と Android スマートフォンを利用

機器として選定した．iPhone 11 で稼働する拡張現実開発環境 ARKit 3 や Android スマートフォンで稼働する同開発環境 ARCore を用いることにより，Zenfone AR と同等レベル以上のリアルな拡張現実表現が可能である．ARKit3 の技術である People Occlusion (CG が人の背後に表示されない機能)を用いるため，暴風雨による飛来物や浸水および土砂崩れの危険性をよりリアルに実感することが可能になった．画面操作によって飛来物の速度や降雨の向きおよび水位や流速、漂流物の有無，土砂崩れの角度を任意に設定することが可能である．

3. AR 暴風雨状況疑似体験アプリ

3. 1　システムの概要

本システムでは，ハードウェアとして，Apple 社製スマートフォン iPhone 11 と紙製ゴーグルを用いる．iPhone 11 にインストールしたアプリにおいて，現在位置における暴風雨発生想定を実風景に重ねて 3D-CG で立体的に表示する．iPhone 11 に紙製ゴーグルを装着することにより没入体験をすることができる．なお，本システムにおけるアプリ（本アプリ）は，iOS 13 がインストールされており，A12 または A13 プロセッサを搭載している iPhone XR，iPhone XS，iPhone 11 Pro，iPhone 11 Pro Max においても動作する．加えて，iPadOS 13 がインストールされており，A12 または A13 プロセッサを搭載している iPad および iPad mini においても動作する．本システムを用いて没入体験を行っている様子を図 10 に示す．

図 10 本システムを用いて没入体験を行っている様子

3. 2 ハードウェア

3. 2. 1　iPhone 11

本研究で使用するスマートフォンは Apple 社製 iPhone 11(iOS13)を用いた．紙製ゴーグルとして，スマホシアターゴーグルクラス　シングル（1 眼タイプ）を用いた．iPhone 11 の外観を図 11，スマホシアターゴーグルクラス　シングル（1 眼タイプ）の外観を図 12 に示す．

図 11 iPhone 11 の外観

図 12 スマホシアターゴーグルクラス　シングル（1 眼タイプ）の外観

3. 3 開発環境

本アプリは Unity2019.2.8f1(64-bit)を用いて開発した.

3. 3. 1 Unity

Unity とは Unity Technologies が開発したゲーム開発用ソフトウェアである. 複数のプラットフォームに対応しており，ウェブプラグイン，PC，各種モバイル端末，ゲーム機向けのコンピュータゲーム開発に用いられる．スクリプト言語として C#，Unity Script(JavaScript)，Boo の 3 種類のプログラミング言語に対応している．3D ゲームを開発することを目的としているため，3D-CG や物理シミュレーションを容易に取り扱うことができる．また，スマートフォンの GPS（全地球測位システム），加速度センサなどの各種センサ情報の取得ができるほか，カメラからリアルタイム映像を取得し，描画することができる．開発したアプリではこれらの機能を用いて，リアルタイム映像に暴風雨や浸水，漂流物の 3D-CG を重ねて表示させている.
本研究では Unity バージョン 2019.2.8f1 を使用した．Unity の実行画面を図 13 に示す.

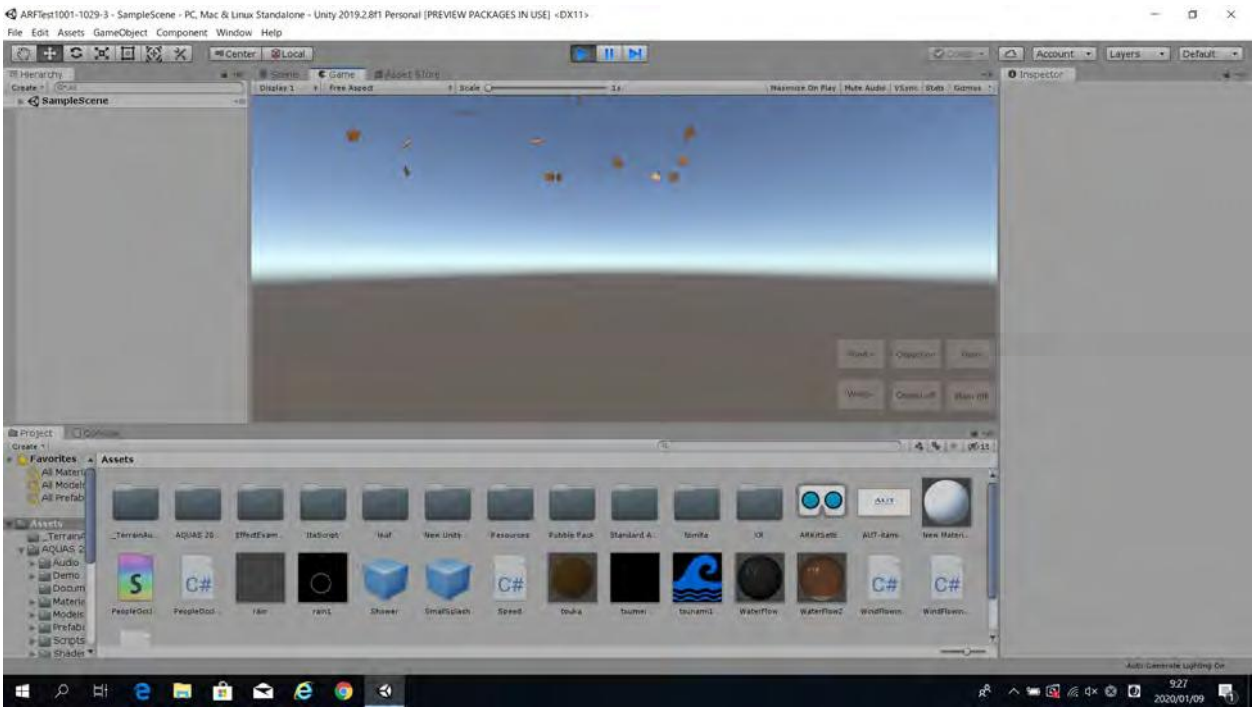

図 13　Unity の実行画面

3. 4　AR アプリ

本研究では，暴風雨発生時の拡張現実(AR)疑似体験アプリを開発した．スマートフォンのカメラで撮影されたリアルタイム映像（現実風景）に3D-CG で表現された暴風雨や水面を重ねて表示できる．iPhone 11では ARKit3を用いることで，よりリアルに CG を現実風景に重ねて表示することが可能である．ARKit3の最先端技術のピープル・オクルージョン(People Occlusion：人の背後に存在する CG が非表示になる機能)を用いる．iPhone 11用アプリの表示画面は左右に分割しないため両眼立体視は行えないが，紙製ゴーグルに装着することで没入体験ができる．Unity で People Occlusion を実装するための処理は大きく分けて以下の3点である[12]．

1) People　Occlusion に必要なテクスチャを集める．
2) AR オブジェクトと人との深度を比較する．
3) ポストエフェクトとして描画する．

ステンシル（型紙：特定領域）とデプス（深度）用のテクスチャは ARHumanBodyManage のプロパティ humanStencilTexture と ARCameraBackground のプロパティ humanDepthTexture で得られる．ARCamera の映像は ARCameraBackground クラスの material プロパティを利用して得られる．

そして人の場所と深度を比較し、必要であれば AR オブジェクトの前面に人の体を描画する．AR オブジェクトと人との深度（スマートフォンからの距離）を比較するシェーダーを図14に示す．

```
float sceneZ = LinearEyeDepth(tex2D(_CameraDepthTexture,　i.　uv));
float delta = saturate(sceneZ - depth);
if (delta > 0.　0)
{
　　return tex2D(_BackgroundTex,　i.　uv);
}
else
{
　　return col;
}
```

図　14 AR オブジェクトと人との深度を比較するシェーダー

最後に CG シーンとカメラ映像を、深度値を元に切り分けてレンダリングする. 図15にポストエフェクトとして描画するシェーダーの重要な箇所を示す.

```
float depth = tex2D(_DepthTex,　uv).　r;
float sceneZ = LinearEyeDepth(tex2D(_CameraDepthTexture,　i.　uv));
float delta = saturate(sceneZ - depth);
if (delta > 0.　0)
{
　　return tex2D(_BackgroundTex,　i.　uv);
}
```

図　15　ポストエフェクトとして描画するシェーダー

降雨の表現については，Unity 標準の Particle システムを使って再現した．本アプリは暴風が右側から発生している状況を想定している．暴風による飛来物の表示例その 1 を図 16，暴風による飛来物の表示例その 2 を図 17 に示す．暴風雨と浸水を同時に表示した例を図 18 に示す．また，雨の表現のキャプチャ画面その 1 を図 19 に，雨の表現のキャプチャ画面その 2 を図 20 に，雨の表現のキャプチャ画面その 3 を図 21 に，物体の出現プログラムを図 22 に，風の風速設定プログラムを図 23 に，物体の挙動のプログラムを図 24 に示す.

図 16　暴風による飛来物の表示例その 1(風は右側から吹いている)

図 16 より，現実風景の人物の背後には飛来物の CG が表示されておらず，奥行き感が

自然に表現されており，リアリティが増していることがわかる．

図 17 暴風による飛来物の表示例その 2(風は右側から吹いている)

図 17 より，現実風景の人物の頭に CG の飛来物が衝突しているかのような表現が行えていることがわかる．

図 18 暴風雨と浸水を同時に表示した例(風は右側から吹いている)

図 18 より，現実風景の人物の輪郭が正確に認識され，人物の背後には CG の水面が表示されていないため，実際に人物が水に浸かっているかのような表現が可能である．建物の壁面も認識し，壁面に CG の水面が表示されていないため，建物が実際に水に浸かっているかのような表現が可能である．水位や水の色は変更可能である．流速も変更可能である．がれきが流れて来る表現も可能である．

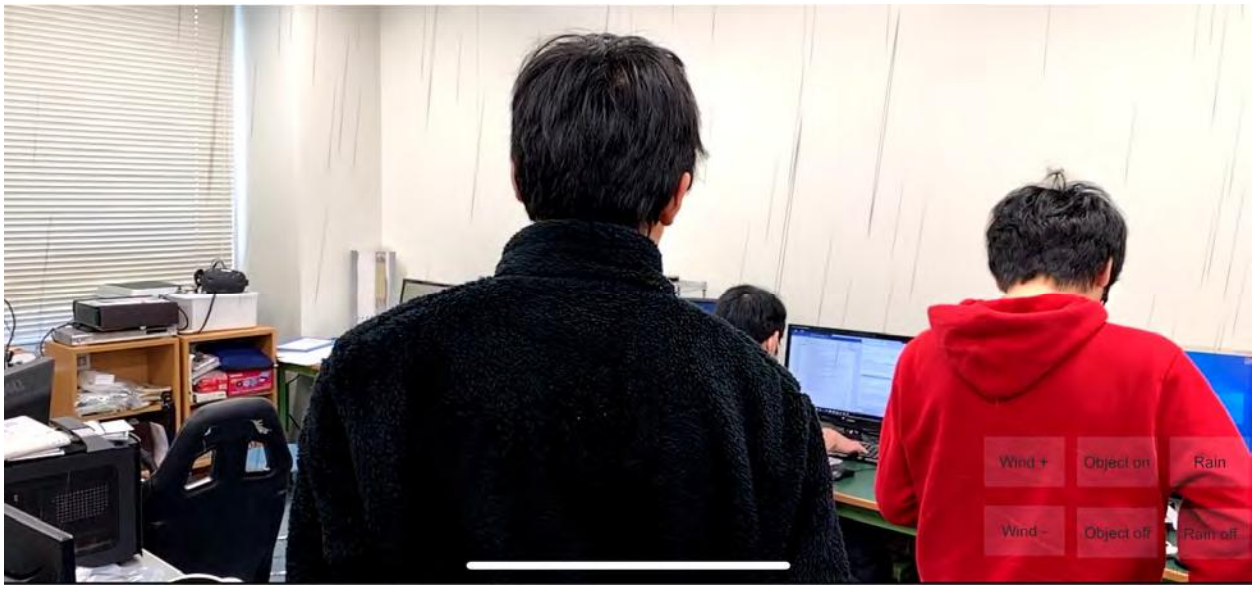

図 19 雨の表現のキャプチャ画面その 1(風は右側から吹いている 風速 1m/s)

図 20 雨の表現のキャプチャ画面その 2(風は右側から吹いている 風速 20m/s)

図 21 雨の表現のキャプチャ画面その 3(風は右側から吹いている 風速 50m/s)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class WindFlowingObjectController : MonoBehaviour
{
    public GameObject box_wood_pre;
    float speed = 1.0f;
    List<GameObject> FollowingObjects = new List<GameObject>();
    float waterLebel = 1.0f;

    // Start is called before the first frame update
    void Start()
    {
        for (int i = 0; i < 25; i++)
        {
            FollowingObjects.Add(Instantiate(box_wood_pre,
             new Vector3(Random.value * Random.Range(-1, 1) * 3.0f + 3.0f, Random.value * 1.0f +
0.7f, Random.value * Random.Range(-1, 1) * 3.0f + 3.0f),
             Quaternion.AngleAxis(90.0f * Random.value, new Vector3(Random.value, Random.value,
Random.value))));
        }

    }

    // Update is called once per frame
    void Update()
    {
        Vector3 pos = new Vector3();
        foreach (GameObject obj in FollowingObjects)
        {
            pos = obj.transform.position;
            pos.x += -0.002f * speed;
            pos.y += -0.0001f*speed;
            if (pos.x <= -5.0f)
            {
                pos = new Vector3(Random.value * Random.Range(-1, 1) * 3.0f + 5.0f, Random.value
* 1.0f + 0.7f, (Random.value * Random.Range(-1, 1) * 3.0f) + 5.0f);

            }
            obj.transform.position = pos;

        }

    }
```

図 22 物体の出現プログラム

図 22 の赤線内のプログラムは，起動時の物体の出現数と，一定範囲内を脱した時に再出現するプログラムである．

```
public void WindPlusDown()
    {
        if (speed < 50) {
            speed += 10f;
        }
    }


    public void WindMinusDown()
    {
        if (speed > 0)
        {
            speed -= 10f;
        }

    }
```

図 23 風の風速設定プログラム

図 23 の赤線内のプログラムは，ボタンを押したとき風速を変更するプログラムである．

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Rotate : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        transform.Rotate(new Vector3(Random.value, Random.value, Random.value));
    }
}
```

図 24 物体の挙動のプログラム

図 24 の赤線内のプログラムは,風に煽られた時の物体の挙動を示すプログラムである．

4. AR 土砂崩れ状況疑似体験アプリ

4. 1　システムの概要

本システムでは, ハードウェアとして, SHARP 社製 Android スマートフォン AQUOS R3 SH-04L を用いる．Google ARCore DepthAPI が動作可能な Android スマートフォンでも動作する．インストールしたアプリにおいて，現在位置における土砂崩れを実風景に重ねて 3D-CG で立体的に表示する．AQUOS R3 SH-04L に紙製ゴーグルを装着することにより没入体験をすることができる．なお，本システムにおけるアプリ（本アプリ）は，Android 9.0 がインストールされている．本システムを用いて没入体験を行っている様子を図 25 に示す.

図 25　本システムを用いて没入体験を行っている様子

4. 2 ハードウェア

4. 2. 1　AQUOS R3 SH-04L

本研究で使用するスマートフォンは SHARP 社製 AQUOS R3 SH-04L（Android 10）を用いた．紙製ゴーグルとして，スマホシアターゴーグルクラス　シングル（1 眼タイプ）を用いた．AQUOS R3 SH-04L の外観（表）を図 26，AQUOS R3 SH-04L の外観（裏）を図 27，スマホシアターゴーグルクラス　シングル（1 眼タイプ）の外観を図 28 に示す.

図 26　AQUOS R3 SH-04L の外観（表）

図 27　AQUOS R3 SH-04L の外観（裏）

図 28　スマホシアターゴーグルクラス　シングル（1 眼タイプ）の外観

4．3 開発環境

本アプリは Unity2018.4.16f1（64-bit）を用いて開発した．

4．3．1 Unity

Unity とは Unity Technologies が開発したゲーム開発用ソフトウェアである．複数のプラットフォームに対応しており，ウェブプラグイン，PC，各種モバイル端末，ゲーム機向けのコンピュータゲーム開発に用いられる．スクリプト言語として C#のプログラミング言語に対応している．3D ゲームを開発することを目的としているため，3D-CG や物理シミュレーションを容易に取り扱うことができる．また，スマートフォンの GPS（全地球測位システム），加速度センサなどの各種センサ情報の取得ができるほか，カメラからリアルタイム映像を取得し，描画することができる．開発したアプリではこれらの機能を用いて，リアルタイム映像に土砂崩れの 3D-CG を重ねて表示させている．

本研究では Unity バージョン 2018.4.16f1 を使用した．Unity の実行画面を図 29 に示す．

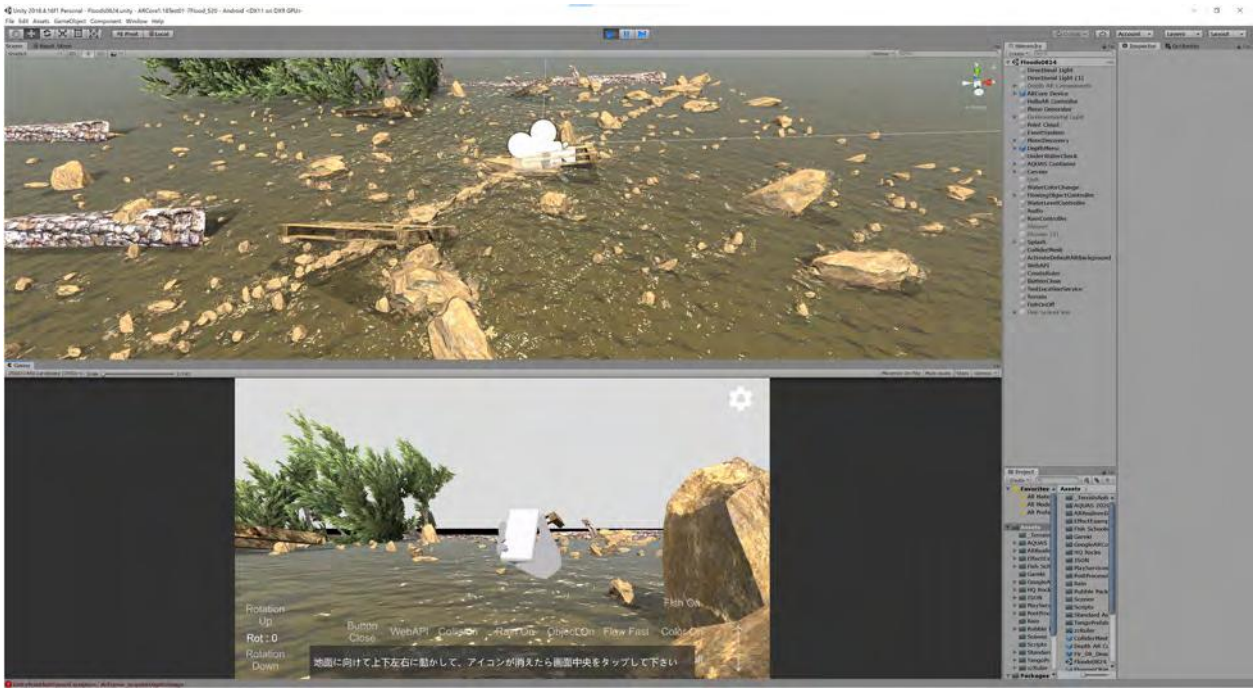

図 29　Unity の実行画面

4．4　AR アプリ

本研究では，土砂崩れ発生時の拡張現実（AR）疑似体験アプリを開発した．スマートフォンのカメラで撮影されたリアルタイム映像（現実風景）に 3D-CG で表現された暴風雨や水面を重ねて表示できる．AQUOS R3 SH-04L では ARCore1.18 を用いることで，よりリアルに CG を現実風景に重ねて表示することが可能である．ARCore1.18 の最先端技術の Depth API を用いる．AQUOS R3 SH-04L 用アプリの表示画面は左右に分割しないため両眼立体視は行えないが，紙製ゴーグルに装着することで没入体験ができる．

図 30 の赤枠内のボタンより，水面の角度を 5 度ずつ変更できる．土砂崩れの表現のキャプチャ画面その 1 （水面の角度 15 度）を図 30 に，土砂崩れの表現のキャプチャ画面その 1 （水面の角度 20 度）を図 31 に，土砂崩れの表現のキャプチャ画面その 1 （水面の角度 30 度）を図 32 に，水面の角度を変更するプログラムを図 33 に，物体を生成するプログラムを図 34 に，物体の挙動のプログラムを図 35 に示す．

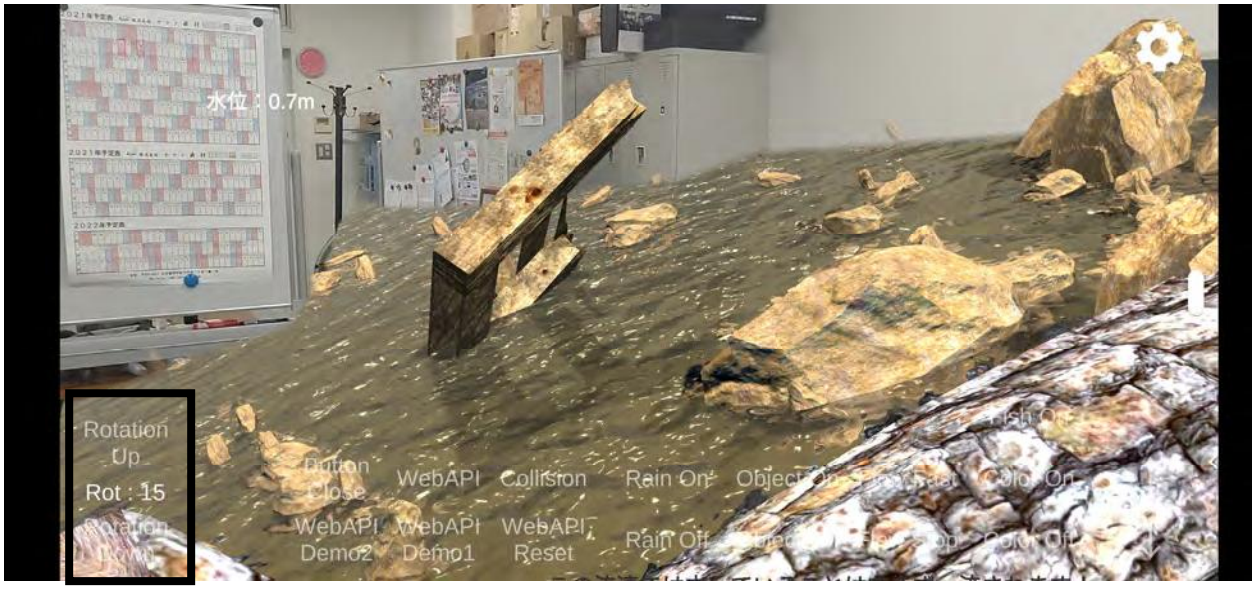

図 30　土砂崩れの表現のキャプチャ画面その 1（水面の角度 15 度）

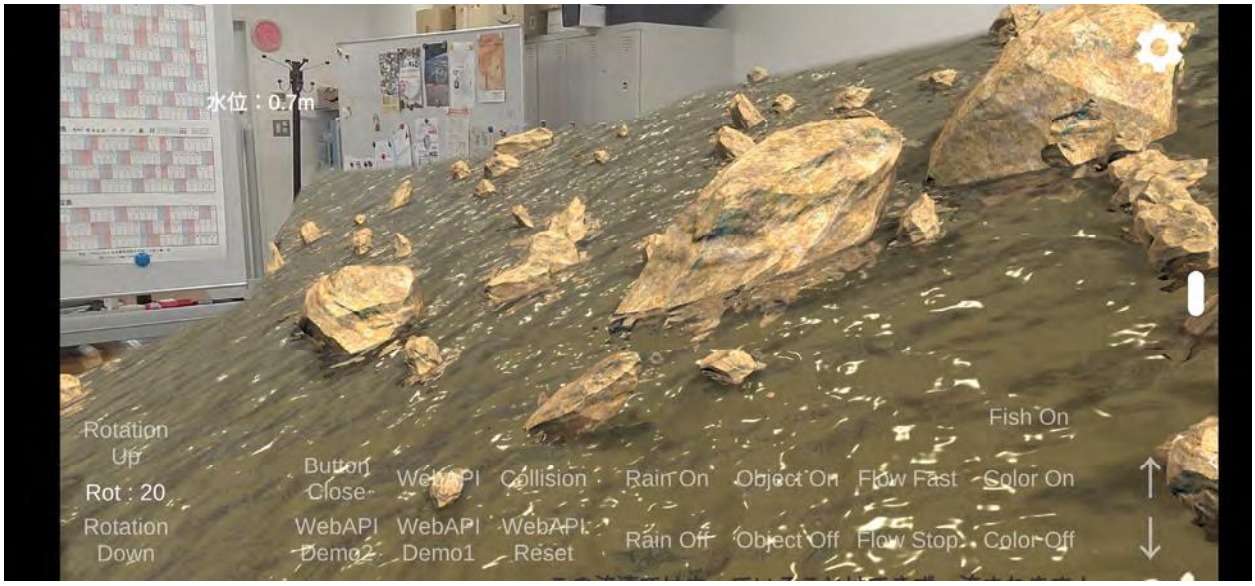

図 31　土砂崩れの表現のキャプチャ画面その 2（水面の角度 20 度）

図 32　土砂崩れの表現のキャプチャ画面その 3（水面の角度 30 度）

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

0 個の参照
public class WaterRotationController : MonoBehaviour
{
    public Transform water;
    public Transform flowingObjectController;
    public Text textRotationNum;
    private int rotationValue = 0;

    0 個の参照
    public void RotationUp()
    {
        if(30 <= rotationValue)
        {
            return;
        }
        water.Rotate(0f,0f,5f);
        flowingObjectController.Rotate(0f,0f,5f);
        rotationValue += 5;
        textRotationNum.text = "Rot : " + rotationValue.ToString();
    }

    0 個の参照
    public void RotationDown()
    {
        if(rotationValue <= 0)
        {
            return;
        }
        water.Rotate(0f,0f,-5f);
        flowingObjectController.Rotate(0f,0f,-5f);
        rotationValue -= 5;
        textRotationNum.text = "Rot : " + rotationValue.ToString();
    }
}
```

図 33　水面の角度を変更するプログラム

図 33 の赤線内のプログラムは，水面の角度を 5 度ずつ変更するプログラムである．

```
// Start is called before the first frame update
0 個の参照
void Start()
{
    for(int i = 0;i < 3;i++)
    {
        FollowingObjects.Add(Instantiate(box_wood_pre,
        createVecterOfInstantiateObjectPosition(),
        Quaternion.AngleAxis(90.0f, new Vector3(0, 0, 1))));  //流木が立たないようにする
        FollowingObjects[FollowingObjects.Count - 1].transform.parent = this.gameObject.transform;
        ResetYAxisPos();
    }

    for(int i = 0;i < 4;i++)
    {
        FollowingObjects.Add(Instantiate(Pallet_pre,
        createVecterOfInstantiateObjectPosition(),
        Quaternion.AngleAxis(90.0f * Random.value, new Vector3(Random.value, Random.value, Random.value))));
        FollowingObjects[FollowingObjects.Count - 1].transform.parent = this.gameObject.transform;
        ResetYAxisPos();
    }

    // HQ Rocks (Big)
    for(int i = 0;i < 40;i++)
    {
        FollowingObjects.Add(Instantiate(hq_rocks[Random.Range(0,hq_rocks.Length - 1)],
         new Vector3(Random.Range(-3f,8f),0f,Random.Range(-3f,8f)),
         Quaternion.AngleAxis(90.0f * Random.value, new Vector3(Random.value, Random.value, Random.value))));
         FollowingObjects[FollowingObjects.Count - 1].transform.parent = this.gameObject.transform;
         ResetYAxisPos();
         // オブジェクトサイズをランダムに変更
         FollowingObjects[FollowingObjects.Count - 1].transform.localScale = new Vector3(Random.Range(0.05f, 0.1f),
                                                                                         Random.Range(0.05f, 0.1f),
                                                                                         Random.Range(0.05f, 0.1f));
    }

    // HQ Rocks(Small)
    for(int i = 0;i < 600;i++)
    {
        FollowingObjects.Add(Instantiate(hq_rocks[Random.Range(0,hq_rocks.Length - 1)],
         new Vector3(Random.Range(-3f,8f),0f,Random.Range(-3f,8f)),
         Quaternion.AngleAxis(90.0f * Random.value, new Vector3(Random.value, Random.value, Random.value))));
         FollowingObjects[FollowingObjects.Count - 1].transform.parent = this.gameObject.transform;
         ResetYAxisPos();
         // オブジェクトサイズをランダムに変更
         FollowingObjects[FollowingObjects.Count - 1].transform.localScale = new Vector3(Random.Range(0.01f, 0.03f),
                                                                                         Random.Range(0.01f, 0.03f),
                                                                                         Random.Range(0.01f, 0.03f));
    }
}
```

図 34　物体を生成するプログラム

図 34 のプログラムは，水面に物体を生成するプログラムである．

```
// Update is called once per frame
0 個の参照
void Update()
{
    Vector3 pos = new Vector3();
    foreach(GameObject obj in FollowingObjects)
    {
        // このスクリプトを含むオブジェクトの向きを基準に移動
        obj.transform.position -= this.transform.forward * speed * Time.deltaTime * 0.2f;
        obj.transform.position -= this.transform.right * speed * Time.deltaTime * 0.2f;  //漂流物の流速
        pos = obj.transform.position;
        if(pos.z <= -3.0f)
        {
            pos = createVecterOfInstantiateObjectPosition();
            pos.y = waterLebel + 0.2f;
        }
        obj.transform.position = pos;
        obj.transform.localPosition = new Vector3(obj.transform.localPosition.x,waterLebel + 0.2f,obj.transform.localPosition.z);
    }
}
```

図 35　物体の挙動のプログラム

図 35 のプログラムは，物体の挙動を示すプログラムである．

5.　評価

本アプリの有用性を評価するために，2 種類のアンケート調査を行った．

5. 1 アンケート調査 1

本アプリの暴風雨状況表現の有用性を評価するために，暴風雨表現がある暴風状況疑似体験アプリと，暴風雨表現のない暴風状況疑似体験アプリを同じ被験者が体験し，体験者にアンケート調査を実施した．20 名が体験した．手順は以下の通りである．

1. 体験者に対し本アプリの仕様の説明を行う．
2. 体験者に降雨表現の有り無しのアプリの片方の体験を行う．
3. 体験者に体験終了後にアンケートを取る．
4. 体験者に 2 で体験しなかったアプリの体験を行う．
5. 体験者に体験終了後にアンケートを取る．

本アプリの暴風雨表現の有り無しの体験順番によってアンケートの回答結果に偏りが出ないように，両アプリの体験順番は体験者によってランダムに設定した．表 1 にアンケート項目を示す．

表 1　アンケート項目

	質問内容	評価
Q1	本アプリを体験して、危機感を感じましたか	1 から 5 の 5 段階評価
Q2	現実に起こった際にどうなるかイメージ出来ましたか	1 から 5 の 5 段階評価
Q3	屋外避難の危険性が実感できましたか	1 から 5 の 5 段階評価
Q4	台風などの暴風雨対策を普段から行おうと思うようになりましたか	1 から 5 の 5 段階評価

暴風雨表現の有り無しのアプリ評価の結果のグラフを図 36 に示す．

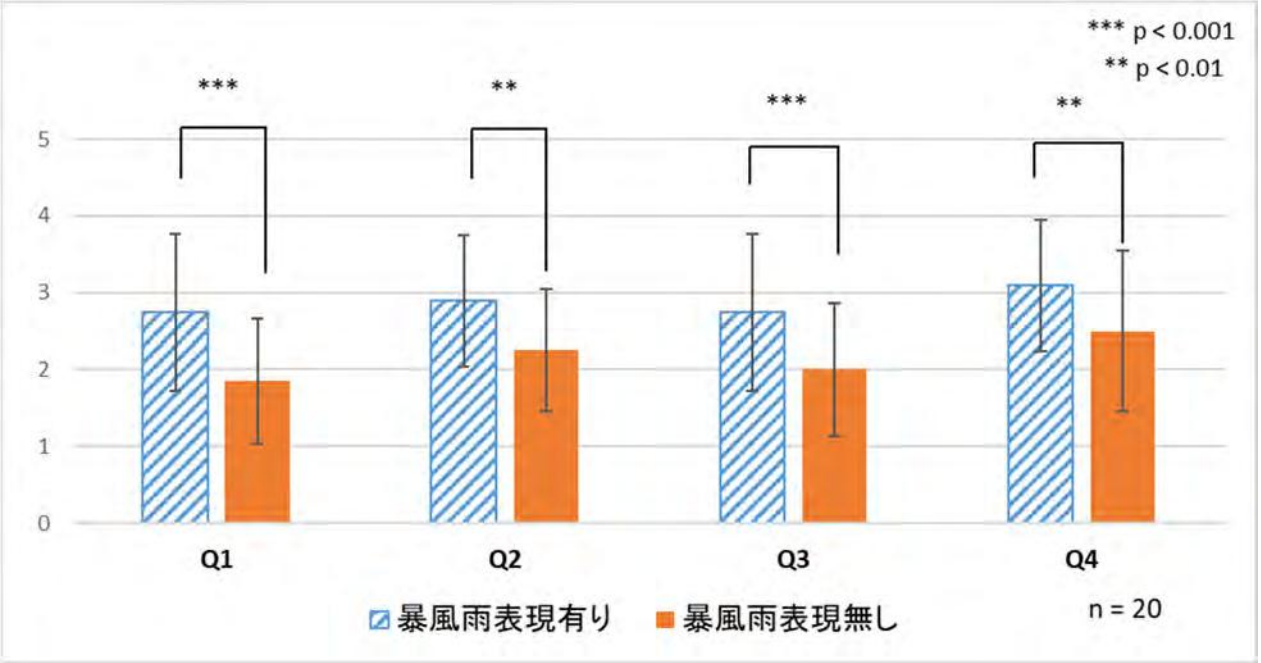

図 36 暴風雨表現の有り無しのアプリ評価の結果

5. 2 アンケート調査 2

本アプリの浸水状況表現の有用性を評価するために，小学校における防災授業にて本アプリの体験会を行い，アンケート調査を行った．2020 年 10 月 5 日に，愛知県蒲郡市立蒲郡北部小学校の 6 年生児童 39 名が体験した．体験後にアンケート調査を行い，39 名から有効回答を得られた．浸水状況体験アプリを児童が体験している様子を図 37～38 に示す．

図 37 浸水状況体験アプリを児童が体験している様子

図 38 浸水状況体験アプリを児童が体験している様子

アンケート調査の質問項目「水害の体験を通して、こわいと思いましたか」において，「すごく思う」が 33 人(85%)，「ややそう思う」が 6 人(15%)，「どちらでもない」「あまりそう思わない」「ぜんぜん思わない」はすべて 0 人(0%)であった．浸水状況体験アプリのアンケート調査結果を図 39 に示す．

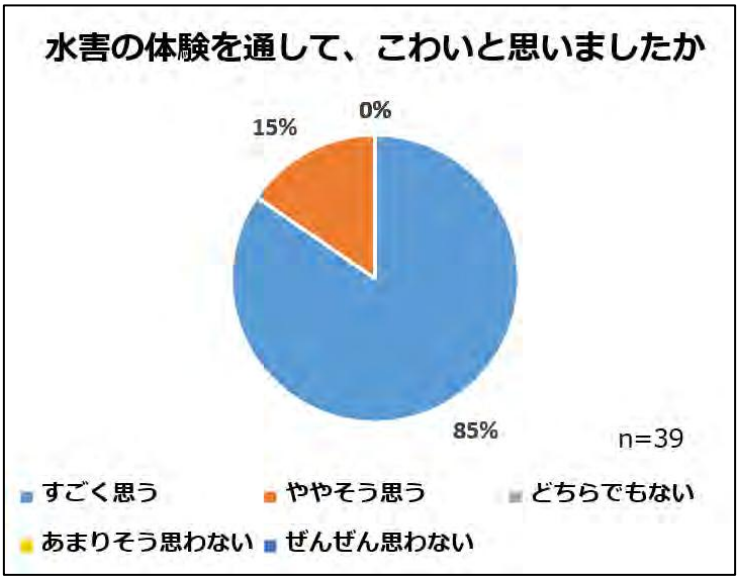

図 39 浸水状況体験アプリのアンケート調査結果

アンケート調査の自由記述欄では，「災害の恐ろしさや怖さを実感できた．しっかり災害に備えようと思った」という旨の記載が多く見受けられた．浸水状況体験アプリのアンケート調査結果（自由記述欄）の例を図 40～42 に示す．

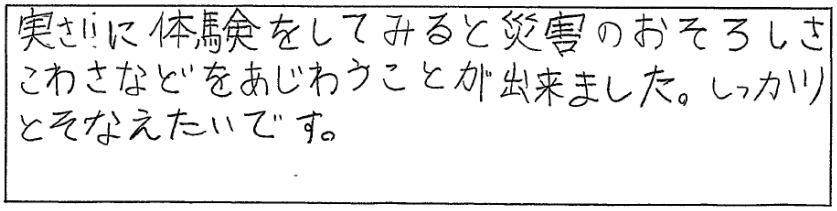

図 40 浸水状況体験アプリのアンケート調査結果（自由記述欄）の例

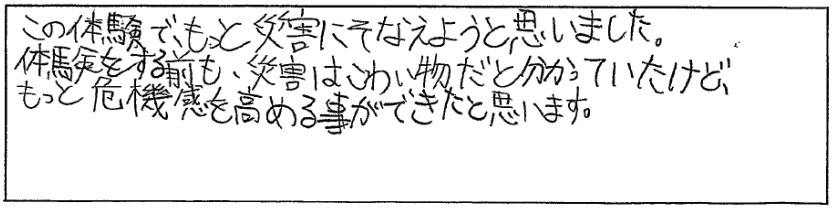

図 41 浸水状況体験アプリのアンケート調査結果（自由記述欄）の例

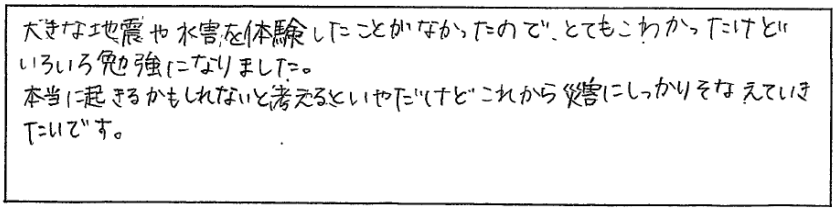

図 42 浸水状況体験アプリのアンケート調査結果（自由記述欄）の例

6. 考察

アンケート調査 1 において「危機感を感じましたか」「現実に起こった際にどうなるかイメージできましたか」「屋外避難の危険性が実感できましたか」「台風などの暴風雨対策を普段から行おうと思いましたか」の 4 つの質問項目すべてにおいて，暴風雨表現が無いアプリと比較して，暴風雨表現が有る本アプリの評価の平均値が高いことから統計的に優位であることが示された．アンケート調査 2 において，「水害の体験を通して、こわいと思いましたか」の質問項目において，「すごく思う」が 85%を占めた．自由記述欄の記載では「災害の恐ろしさや怖さを実感できた．しっかり災害に備えようと思った」という旨の記載が多く見受けられた．これらのことから，本アプリの有用性が示された．

7. 結論

本研究では，平時における風水害への危機意識の向上を目的として，一般的に入手が容易な Apple iPhone 11 と Android スマートフォンを用いて，暴風雨時と土砂崩れにおける災害発生状況を現実風景に重ねて CG（コンピュータ・グラフィックス）表示し，疑似体験できる拡張現実(AR)スマートフォンアプリを開発した．一般的に普及しているスマートフォンを用いるため汎用性が高く複数人同時での体験が可能であり，避難訓練において幅広く活用できた．拡張現実アプリ開発環境 Apple ARKit3 や Google ARCore DepthAPI の機能を活用し，人物や家具などの形状を自動認識し，暴風による飛来物の落下状況や豪雨の降り方や浸水発生および土砂崩れの状況を一般的に普及しているスマートフォンにおいてリアルに表現することが可能になった．そのため，本アプリの体験者は風水害による災害のリスクを「自分のこと」として実感できるようになった．画面操作によって飛来物の速度や降雨の向きおよび水位・流速や漂流物の有無や土砂崩れの角度を設定することを可能にした．本アプリの評価において「危機感を感じましたか」

「現実に起こった際にどうなるかイメージできましたか」「屋外避難の危険性が実感できましたか」「台風などの暴風雨対策を普段から行おうと思いましたか」の全ての評価が高く，本システムは危機意識の向上に有用であることが示された．今後は，東三河地域をはじめとする各地の防災訓練や防災イベントにおいて本アプリを実用してアンケート調査を実施し，有用性の検証を重ねていく．また，体験者や防災の専門家から多くの意見を収集し，機能の追加を行いアプリの防災教育効果を高めていく．また，本システムを活用した防災授業を教材としてパッケージ化し，全国の教育機関に展開していく．

参考文献リスト

[1] 内閣府：令和元年台風第 19 号に係る被害状況等について，http://www.bousai.go.jp/updates/r1typhoon19/index.html（アクセス日：2020 年 1 月 8 日）

[2] YAHOO!ニュース：甚大な被害をもたらした台風 19 号　暴風による被害を検証，https://headlines.yahoo.co.jp/hl?a=20191108-00011693-weather-soci（アクセス日：2020 年 1 月 8 日）

[3] 内閣府：防災に関する世論調査，2017.

[4] 鈴木康弘：防災・減災につなげるハザードマップの活かし方，2015.

[5] 細川直史：拡張現実(AR)を用いたリスクコミュニケーションツールの開発，電子情報通信学会総合大会講演論文集 基礎・境界講演論文集，S-64，2012.

[6] 細川直史：消防防災分野における拡張現実の活用，映像情報メディア学会誌,Vol. 66,No. 11，pp.929-933，2012.

[7] 総務省情報通信政策研究所：平成 26 年情報通信メディアの利用時間と情報行動に関する調査，2015.

[8] リスク対策．com．　バーチャルリアリティを生かした防災訓練，http://www.risktaisaku.com/articles/-/875　（アクセス日：2020 年 1 月 8 日）

[9] 東京消防庁　＜組織・施設＞＜VR 防災体験車の概要＞ https://www.tfd.metro.tokyo.lg.jp/ts/bousai_fukyu/　（アクセス日：2020 年 1 月 27 日）

[10]　iDEA CLOUD 防災 VR 台風・暴風雨編を開発・パッケージ化し 2019 年 11 月 4 日（月）にリリース予定, https://ideacloud.co.jp/blog/works/bousai-vr-taifuu.html (アクセス日：2020 年 1 月 8 日)

[11]　板宮朋基，吉村達之：複合現実による災害想定没入体験アプリ Disaster Scope の開発と避難訓練における活用. 日本災害情報学会論文誌 災害情報, No.16-2, 191-198, 2018.

[12]　Unity AR Foundation で People Occlusion をやってみる，http://edom18.hateblo.jp/entry/2019/08/11/223803（アクセス日：2020 年 1 月 04 日）