総合学園ヒューマンアカデミー横浜校
ゲームカレッジ研究生

角屋 潤

PORTFOLIO

ポートフォリオ

・プロフィール

氏名：角屋 潤 （２０）
希望職種：プログラマー
プログラミング歴：2年

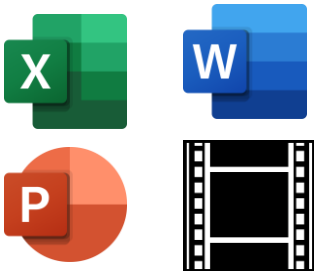
所持スキル

言語

開発ツール

開発支援ツール

その他ツール



C、 C++、 C#、 PHP、 JavaScript、 ActionScript3.0
VisualStudio、 Unity
git、 Adobe Animate、 Adobe illustrator、
Adobe Photoshop、 Xampp、 FileZilla
Excel、 Word、 PowerPoint、 AviUtl

・アピールポイント

私はゲームコンテストへ応募するゲームのチーム制作にて、リードプログラマーを、3度務めた経験があります。

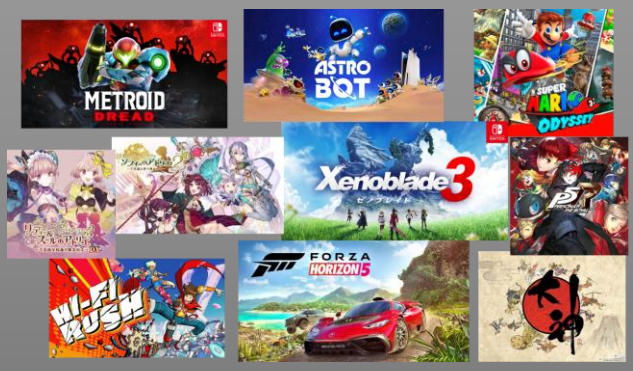
またチーム制作で使用したUnityのほかにも、Direct3D11を用いた3Dプログラミングを、技能五輪のウェブデザイン職種に出場した経験から、ウェブやサーバーサイドの技術も習得しています。

様々な経験をするなかで、新しい技術を学ぶことの面白さに気づくことができ、成長し続けたいと考えています。

・好きなゲーム

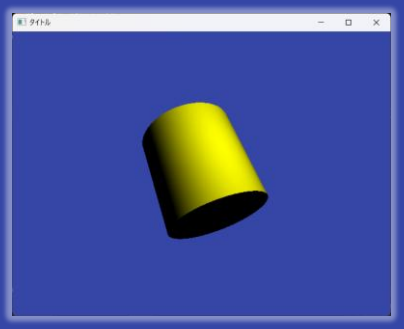
特にお気に入りの10本

ジャンルはアクションやRPGが特に好きですが、他のジャンルなども幅広くプレイしています。

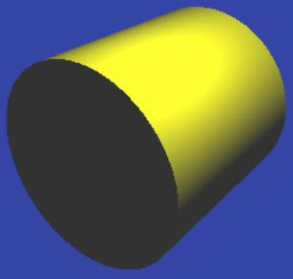
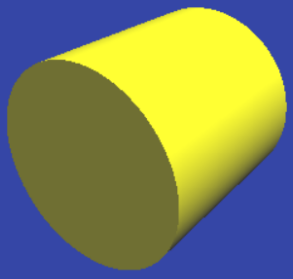
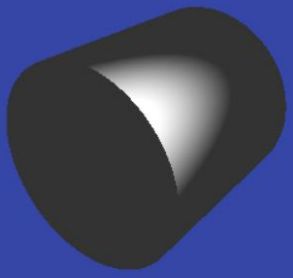
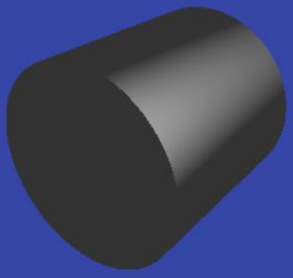
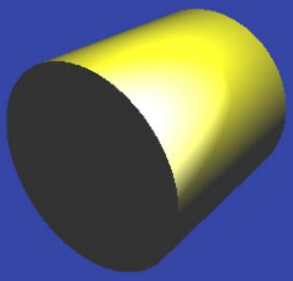
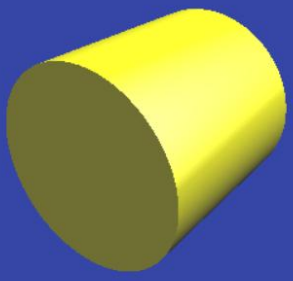


DirectX 個人製作

Direct3D11(HLSL)でのライティング



シェーダープログラムを習得するために制作したphong反射のデモです。
こちらはVisual Studioを用いてフルスクラッチで開発をしています。

	点光源	平行光源
Diffuseのみ		
Specularのみ		
Diffuse+Specular		

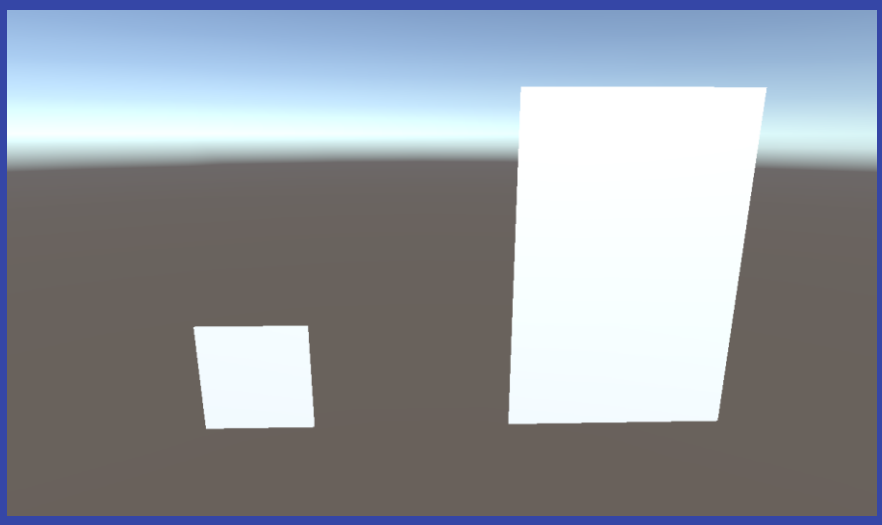
FBX形式 3Dモデル生成 ツール QuadGenerator

３Dツールを開かずにサンプルデータを作成できるようにするために制作した、コマンドラインから幅と高さを指定して、四角形の板のFBX形式で出力するツールです。
Autodesk FBX SDKを利用しています。

```
C:\Study\Apps>QuadGenerator --help
[--size-x <value>]  Quadの横のサイズを数値で指定
[--size-y <value>]  Quadの縦のサイズを数値で指定
[--mapping <value>] 法線情報を保存する方式(EMappingMode)を数値で指定
        1:頂点情報に直接設定(eByControlPoint)
        2:Polygonを作っているそれぞれの頂点に設定(eByPolygonVertex)
        3:Polygonに直接設定(eByPolygon)
        4:全てを同一のものを設定(eAllSame)
[--reference <value>] 法線情報を頂点情報に関連付ける方式(EReferenceMode)を数値で指定
        1:直接指定(eDirect)
        2:配列からIndexを使い指定(eIndexToDirect)
[<name>] 出力ファイルの名前を指定
[-h | --help] ヘルプを表示
```

```
width: 1
height: 1
fileName: quad1
quad1を出力しました。
```

```
width: 2
height: 3
fileName: quad2
quad2を出力しました。
```



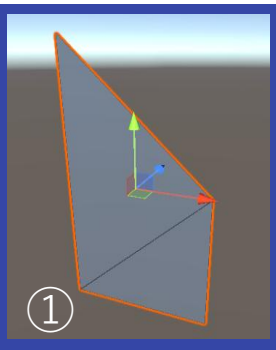
↑Unity上での表示

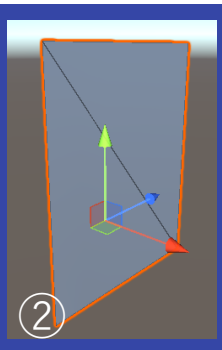
FBX形式での3Dツールの書き出しとUnityの読み込みの調査結果

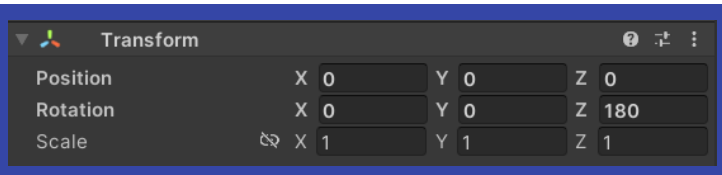
QuadGeneratorの制作において、Unityで使うことを想定したFBXファイルを市販の3Dツールがどのような設定で出力しているか、またそのデータをUnityがどのように解釈して表示しているかを知る必要がありました。

```
const auto scene = FbxScene::Create(manager, "SceneName");
FbxAxisSystem unityAxis(
    FbxAxisSystem::EUpVector::eYAxis,
    static_cast<FbxAxisSystem::EFrontVector>(-FbxAxisSystem::EFrontVector::eParityOdd),
    FbxAxisSystem::ECoordSystem::eLeftHanded);
scene->GetGlobalSettings().SetAxisSystem(unityAxis);
```

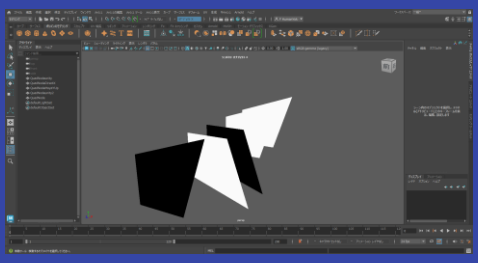
FBXのシーンには座標系の設定が存在します。ここで正しく設定しなくてはいけません。上記の設定はUnityのYを上とする左手座標系の設定です。これで出力した結果が以下のようになります。





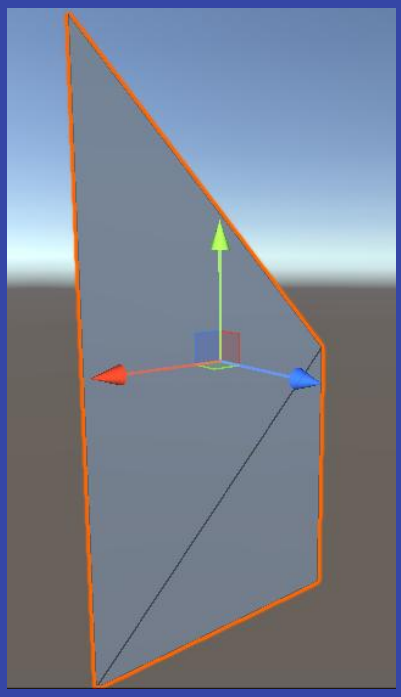


①を想定して座標情報をもたせましたが②のようにローテーションがかかっており、また左右反転しています。



そこで私はMayaとBlenderで同じ形のFBXを出力し、どんな頂点座標をもち、どんな座標系で出力しているかを調査しました。

検証の結果、MayaやBlenderからUnity用の出力設定で出力したFBXはどれも右手座標系を表す設定になっていました。



このことから、UnityはFBXを右手座標系のものとしてインポートし、右手から左手に変換するため、X座標のみを反転させ扱っている、と私は考えました。
様々なパターンで検証しましたが反例はありませんでした。

なので、出力時の適切な設定は、右手座標系のYupである、MAYAのYupの設定を使用し、頂点などの座標は右手座標系で考えて入力するのが、正しいものと考えられます。

```
const auto scene = FbxScene::Create(manager, "SceneName");
scene->GetGlobalSettings().SetAxisSystem(FbxAxisSystem::EPreDefinedAxisSystem::eMayaYUp);
```

Unityがこの仕様になっている理由は、３Dソフトの多くが右手座標系を採用していることや、Zのプラス方向にステージを作るのが一般的なため、反対を向くのはむしろ都合がいいと考えているからなのではないかと、私は考えています。

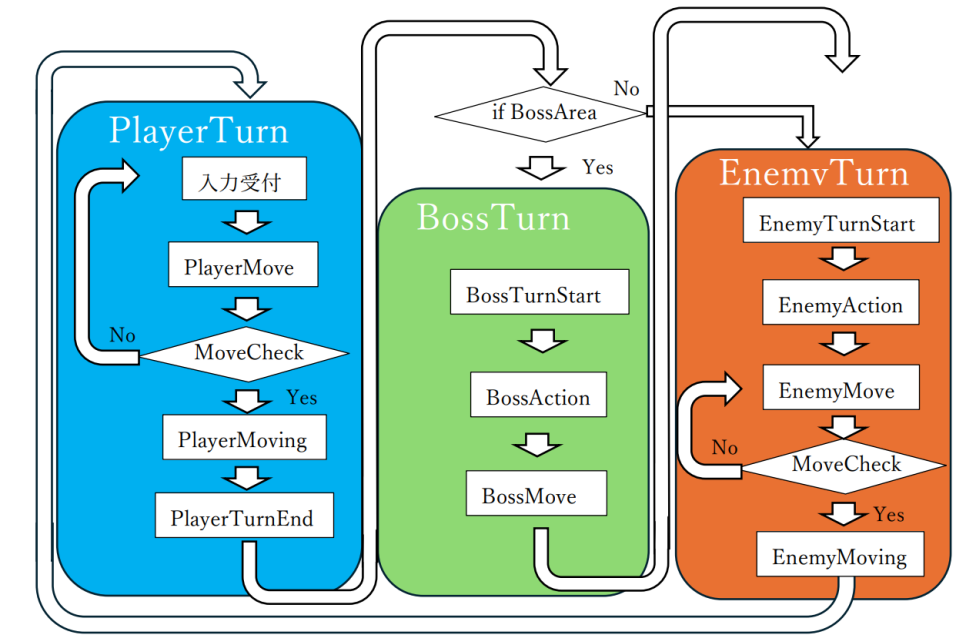
滅算勇者

げんざんゆうしゃ

このゲームは、敵を倒すとレベルが下がってしまう勇者を操作して、ステージ最奥のボスを倒すことが目的のダンジョンRPGです。

ジャンル： 2.5Dダンジョンrpg
制作人数： プログラマー3人
プランナー2人
CGデザイナー2人
制作期間： 3か月
担当箇所： ゲームプレイ部分全般、敵のレベルやボスのHPのUI、エフェクト

この制作は3度目の制作で、初めてアクションゲームではないゲームの制作でした。
このゲームは、一回移動することで敵も一回アクションを起こす、ターン制のゲームです。
今まで以上に呼ばれる関数の流れを把握することが大切だったので、その流れを図にして表すなど、工夫をしながら制作を進めました。



敵の移動のルート探索

敵の移動には、**A*アルゴリズム**によるルート探索を行っており、プレイヤーに向かって最短ルートで近づいてきます。木や他の敵などの障害物がありますが、それは迂回して進み、完全に道がふさがれていたら移動をしません。↓

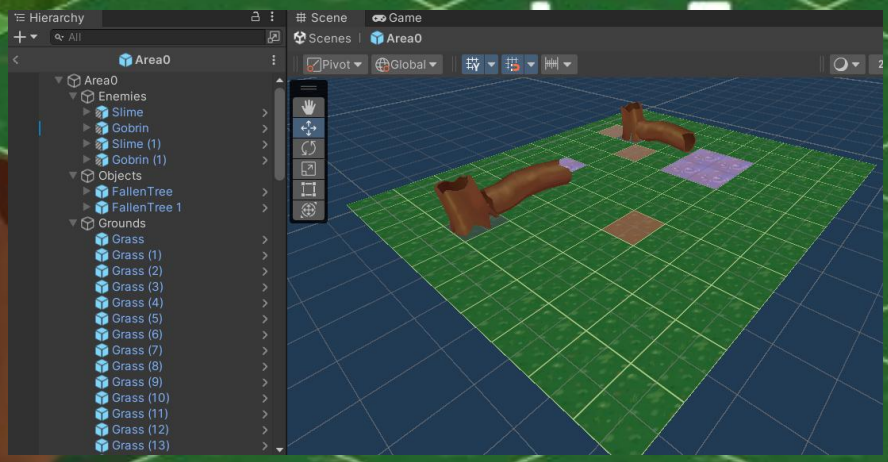
マス目分のNodeクラスがあり、それぞれにステート、コスト、親ノードを設定していき、最終的に親ノードを辿って移動方向を導き出します。

```
using UnityEngine;

namespace Genzan
{
    public class Node
    {
        public Vector2Int position = new(0, 0);
        public enum State { None , Open , Closed , Lock};
        public State state = State.None;
        public float gCost = 0;
        public float hCost = 0;
        public float FCost => gCost + hCost;
        public Node parentNode;
    }
}
```

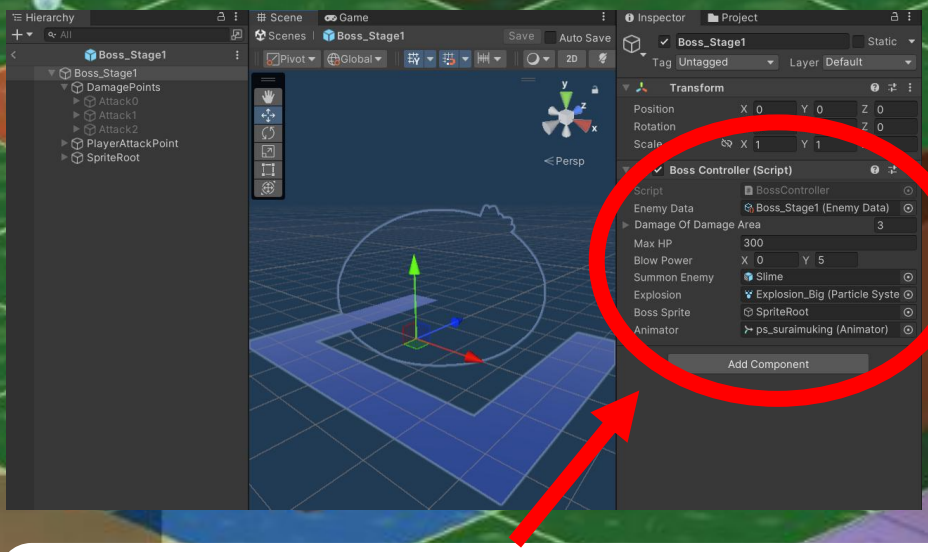
```
//探索成功時終了処理
finish:
//親ノードを辿り最初のノードまで戻る
while (true)
{
    //最初のノードまで戻ったら、移動方向を求める
    if(baseNode.parentNode.position == start)
    {
        int moveDirection = DirectionCal(baseNode.parentNode.position,baseNode.position);
        return moveDirection;
    }
    baseNode = baseNode.parentNode;
}
```

マップクリエイト

プランナーがレベルデザインをしやすくするために、ステージの地面やオブジェクト、敵の配置といったものを簡単に変更することができるようにしました。例えば敵を追加したければEnemies配下に配置するだけで機能します。またほかにもボスの攻撃範囲や、ボスへの攻撃が可能なマスも同様に機能するようにしました。これらはスクリプトで子要素を取得することで実装しています。



Lv.5



他にもステータスの変更も基本、インスペクターから行えるようにしました。

プレイヤーの連続移動

敵がいない場合などに、長押ししていれば移動が中断されずに連続移動できるようにしました。
仕組みとしては、移動終了１ｆ後にプレイヤーのターンに戻ってきていて、入力があるかで判定しています

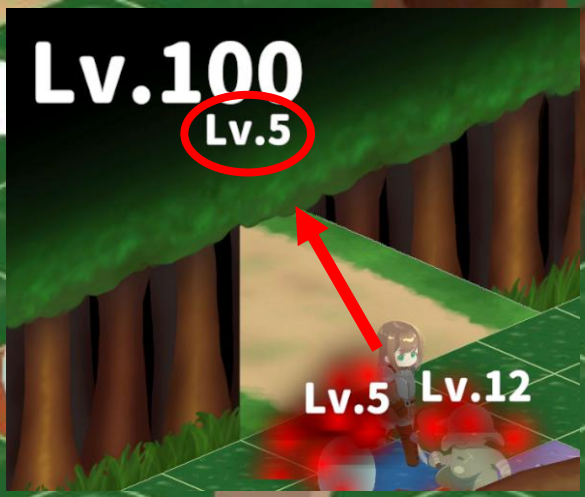
```
//移動終了時ボタンを押し続けているかチェック
IEnumerator HoldCheck()
{
    yield return null;
    //1フレーム後にプレイヤーのターンであるかチェック
    if(gM.turnState == GameManager.TurnState.PlayerTurn)
    {
        bool holding = false;
        for (int i = 0; i < inputDirection.Length; i++)
        {
            if (inputDirection[i])
            {
                holding = true;
            }
        }
        //移動のボタンを押していなかったらアニメーション停止
        if (holding == false)
        {
            StopAnimation();
        }
    }
    else
    {
        StopAnimation();
    }
}
```

キャラの移動のチェック

プレイヤーや敵の移動では、移動先のマスにオブジェクトがないかをチェックしています。
プレイヤーの場合はエリア外、敵は別の敵の位置にも移動できなくさせています。

```
//移動可能かチェック
private bool MoveCheck(int directionNumber)
{
    Vector2Int nextPosition = Position;
    //次の位置を移動方向から計算
    switch (directionNumber)...

    //移動先にオブジェクトがないかチェック
    for (int i = 0; i < gM._object.Count; i++)
    {
        if (nextPosition == gM._object[i].position)
        {
            return false;
        }
    }
    //移動先に別の敵がいないかチェック
    for(int i = 0; i < gM.enemy.Count; i++)
    {
        if(nextPosition == gM.enemy[i].controller.Position
          && gM.enemy[i].controller.active
          && gM.enemy[i].gameObject != this.gameObject)
        {
            return false;
        }
    }
    return true;
}
```

レベルのUIの移動



敵の頭上のレベルを表すUIは、敵の位置からカメラ上の座標を求め、その位置に表示しています。
その敵が倒された際には、画面左上のプレイヤーのレベルのUIの方向に向けて移動させていき、重なったタイミングで数値を変化させています。
毒沼や宝箱などでは、効果が表れたときに、レベル表示のUIを生成します。

```
//生成
GameObject eL = Instantiate(enemyLevel);
eL.transform.SetParent(this.transform, false);
enemyLevelText.Add(eL);
activeEnemyNumber.Add(i);
//敵のワールド座標を画面上の座標に変換
Transform transform = gM.enemy[i].gameObject.transform;
Vector3 worldPosition = transform.position;
Vector2 screenPosition = camera.WorldToScreenPoint(worldPosition);
screenPosition += shift;
RectTransform rectTransform = eL.GetComponent<RectTransform>();
rectTransform.localPosition = screenPosition;
TextMeshProUGUI textMeshProUGUI = eL.GetComponent<TextMeshProUGUI>();
textMeshProUGUI.text = "Lv." + gM.enemy[i].level.ToString();

moveTimer.Add(0);
```

ボスの行動

ボスは他の敵とは全く別の処理を行っています。攻撃準備のCharge、実際に攻撃するAttack、プレイヤーに攻撃された際プレイヤーを引き飛ばすBlow、の3つの状態があります。

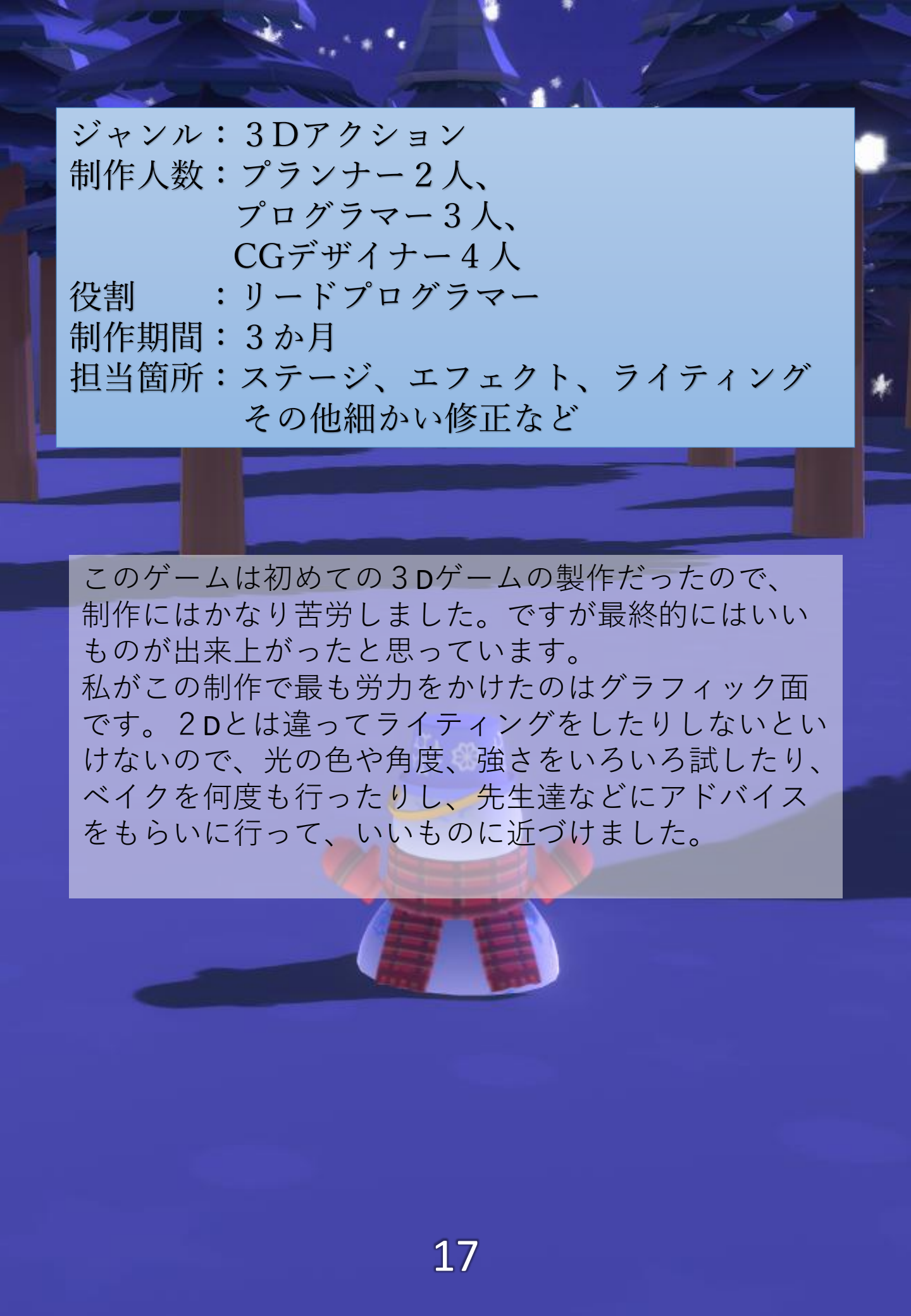
Chargeでは、ランダムな値から攻撃する技を決定し攻撃範囲を表示、
Attackではその攻撃範囲にプレイヤーがいるかどうか、いたらプレイヤーのダメージ判定を、
Blowでは、攻撃のキャンセルとプレイヤーの位置変更を行います。

Charge

Attack

Blow

このゲームは、雪だるまを転がして大きくしながらゴールである家を目指す、アクションゲームです。



ジャンル：３Ｄアクション
制作人数：プランナー２人、
　　プログラマー３人、
　　CGデザイナー４人
役割　　：リードプログラマー
制作期間：３か月
担当箇所：ステージ、エフェクト、ライティング
　　その他細かい修正など

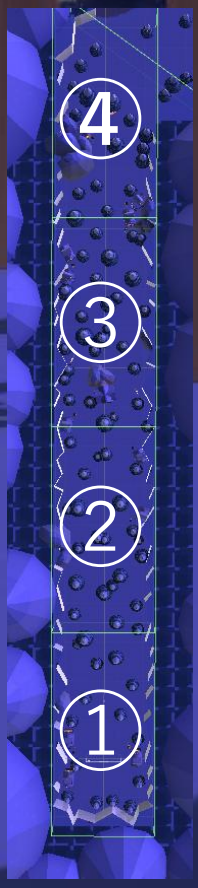
このゲームは初めての３Dゲームの製作だったので、制作にはかなり苦労しました。ですが最終的にはいいものが出来上がったと思っています。
私がこの制作で最も労力をかけたのはグラフィック面です。２Dとは違ってライティングをしたりしないといけないので、光の色や角度、強さをいろいろ試したり、ベイクを何度も行ったりし、先生達などにアドバイスをもらいに行って、いいものに近づけました。

ゲームに合った絵作り

雪の降る冬の夜を表現するために、UnityのGI等を利用するなど、さまざまな工夫をしました。

Particle

ゲームのシーンでは雪のパーティクルを上から降らしています。パフォーマンス下げる要因になると考え、プレイヤーの周りにだけ降らしています。例えば②にいるときは①②③に、③にいるときは②③④だけ降らせます。



Bloom

雰囲気づくりのため、Unityの**Post-process**を利用しブルームをかけています。

CSVデータを読み込みマップを生成するEditor拡張

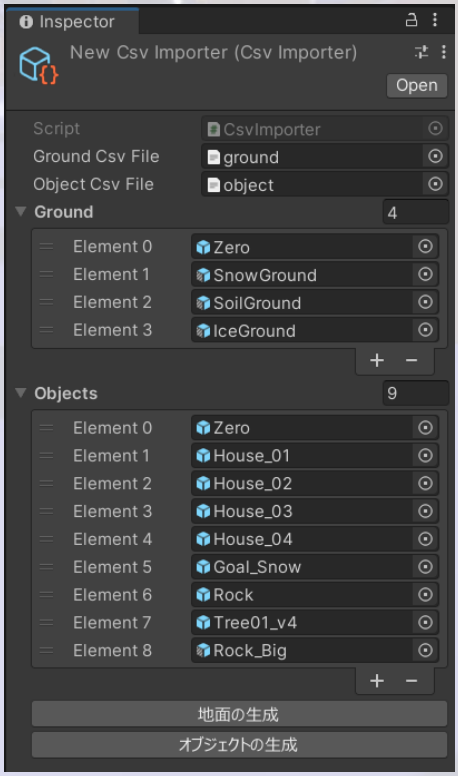
数値を入れたCSVデータを読み込み、対応した地面やオブジェクトを配置できるEditor拡張を作成しました。

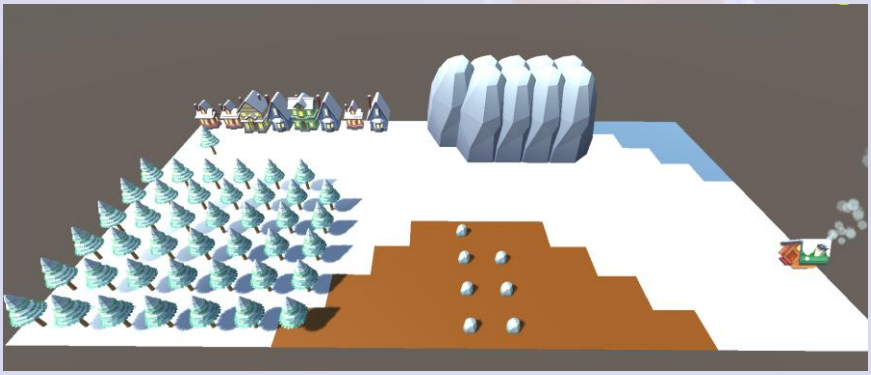
ground.csv

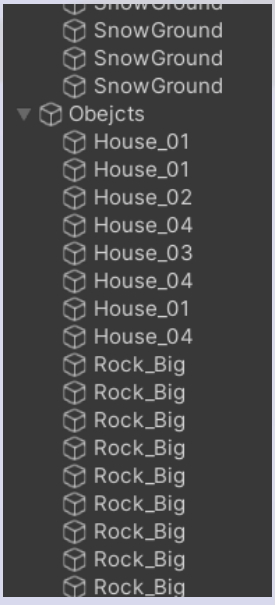
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3	3	3	3
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3	3	3
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3	3
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	2	2	2	2	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	2	2	2	2	2	2	1	1	1	1	1	1
9	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	1	1	1	1	1
10	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	1	1	1	1

object.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	1	1	2	4	3	4	1	4	0	0	8	8	8	8	8	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	8	8	8	8	0	0	0	0	0
3	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	7	7	7	7	7	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	7	7	7	7	7	7	7	0	0	0	0	0	0	0	0	0	0	0	0	0
7	7	7	7	7	7	7	7	0	0	0	6	0	0	0	0	0	0	0	0	0
8	7	7	7	7	7	7	7	0	0	0	6	6	0	0	0	0	0	0	0	5
9	7	7	7	7	7	7	7	0	0	0	6	6	0	0	0	0	0	0	0	0
10	7	7	7	7	7	7	7	0	0	0	6	6	0	0	0	0	0	0	0	0







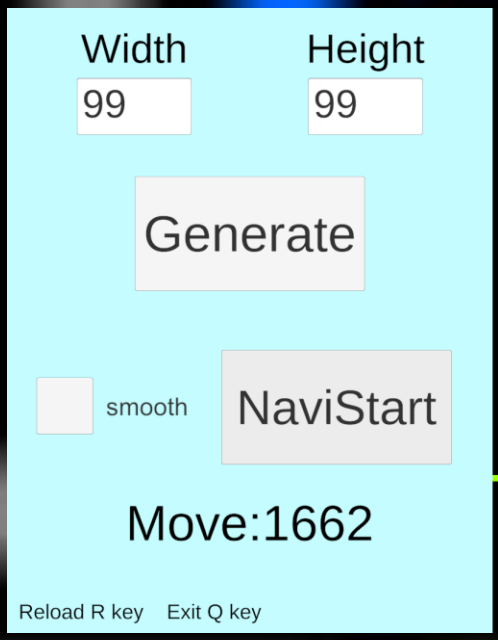
このゲームは釣りパートと魚パートがあり、釣りパートのミニゲームで釣り上げた魚を、魚パートで操作しできるだけ遠くに、かついい場所に飛ばすゲームです。

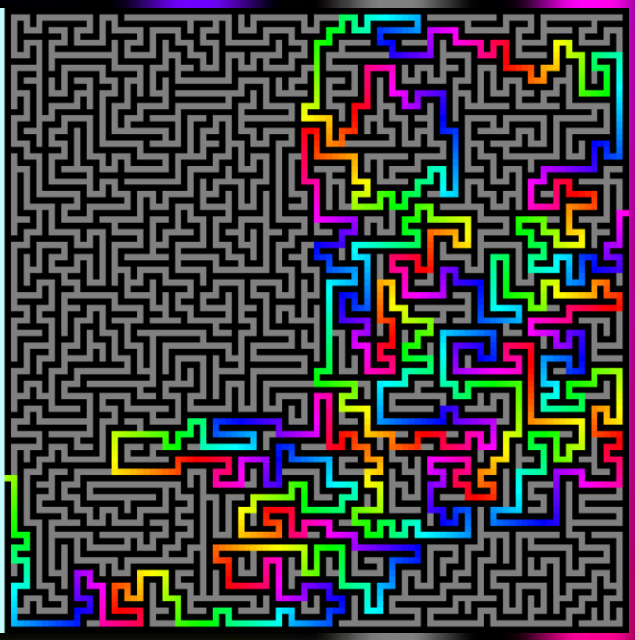
ジャンル：２Ｄフィッシング＆アクション
制作人数：プランナー１人、
　　プログラマー２人、
　　CGデザイナー２人
役割　　：リードプログラマー
制作期間：２週間
担当箇所：Git管理、魚パートのプレイヤー操作、
　　ステージギミック、ステージ生成、UI、
　　ポーズ画面、リザルト画面、SE、BGM

このゲームの制作が初めてチームで作ったゲームです。本制作で私はプログラマーをまとめる立ち回りをしました。担当箇所を明確にし、Gitの競合問題を起こさないよう取り組みました。ほかにも制作期間内に実現可能かの判断や、発注する画像の適切なサイズの判断など、プランナーとCGデザイナーとの連携も積極的に行いました。このゲームは東京ゲームショウで展示をしたのですが、主に子供に受けがよく、たくさん遊んでもらえました。

MazeSearch

制作人数：1人
制作期間：2日





この作品は入力した縦横のマス数で迷路を自動生成するツールです。また生成した迷路のスタートからゴールまでの道順を示すナビゲーションを表示させることも可能です。

迷路の自動生成

迷路は穴掘り法というアルゴリズムを使い実装しています。これは最初に基準点を作り、そこから穴を掘るように道を作っていく方法で、最後に上辺または左辺にスタートを、下辺または右辺にゴールを設定しています。

```
//指定の方向に掘ることができるかチェックし掘る
for (int i = 0; i < digDirection.Count; i++)
{
    bool canMove = false;
    switch (digDirection[i])
    {
        case CNum.UP:
            if (baseBlock.position.y + 2 >= 0 && baseY + 2 < height)
            {
                if (block[baseX, baseY + 2].dug == false)
                {
                    block[baseX, baseY + 1].dug = true;
                    block[baseX, baseY + 2].dug = true;
                    canMove = true;
                    block[baseX, baseY].digDirection = i;
                    block[baseX, baseY + 2].parent = block[baseX, baseY];
                    baseBlock = block[baseX, baseY + 2];
                }
            }
            break;
```

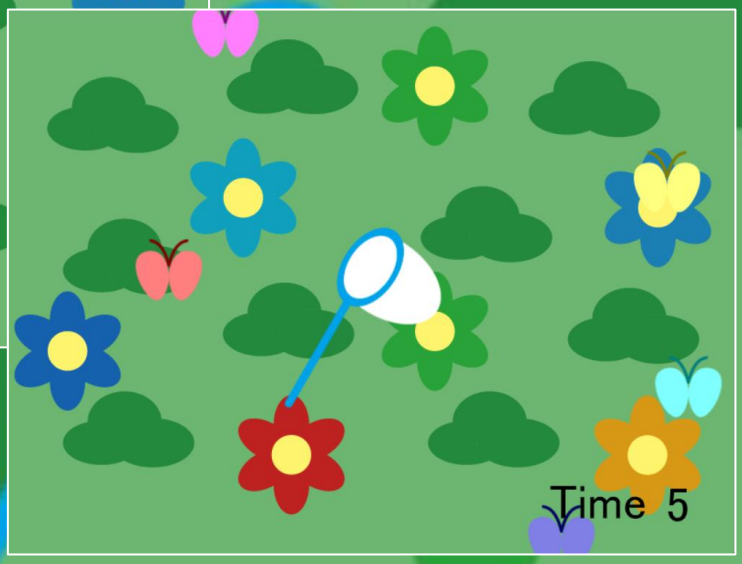
スタートからゴールまでのナビゲーション

ナビゲーションにはA*アルゴリズムを使いスタートからゴールまでのルートを導き出しています。開始座標と終了座標と移動可能な座標データを使い、スタートからの移動方向の配列を受け取ります。そしてその移動方向をもとに、進むマスの色を変えていきルートを表示します。

```
public IEnumerator MoveNavi()
{
    int firstColor = Random.Range(0, 100);
    mM.mazeBlock[mM.startBlock.position.x, mM.startBlock.position.y].GetComponent<Image>().color
        = Color.HSVToRGB((float)(moveCount + firstColor) / 100 % 1, 1, 1);
    Vector2Int position = mM.startBlock.position;
    PathFinding pathFinding = new();
    List<int> moveDir
        = pathFinding.FindPath(mM.startBlock.position, mM.goalBlock.position, mM.Movable(),mM.Width,mM.Height);
```

Butterfly Catch

ジャンル：虫取りゲーム
制作人数：1人
制作期間：4日



このゲームは自由に動く蝶たちを虫取り網で捕まえていき、すべて捕まえるとクリアとなる虫取りゲームです。
この蝶は虫取り網が近づくと逃げていきます。

蝶の動きをコントロールするAI

この蝶は常に虫取り網との距離と角度を求めています。

```
//蝶の動きのコントロール
stage.addEventListener(Event.ENTER_FRAME,butterflyControl);

function butterflyControl(evt:Event) :void{
    for(var i = 0; i < Butterfly.length; i++){
        if(ButterflyActive[i]){
            //角度と距離の計算
            Angle[i] = Math.atan2(-Net_mc.y - -Butterfly[i].y,Net_mc.x - Butterfly[i].x) * (180 / Math.PI);
            var a = (Butterfly[i].x - Net_mc.x);
            var b = (Butterfly[i].y - Net_mc.y);
            Distance[i] = Math.sqrt(a * a + b * b) - ButterflyRadius - NetRadius;
```

そしてその距離に応じて、自由に動くか、
虫取り網から逃げるかの2つの動きをします。

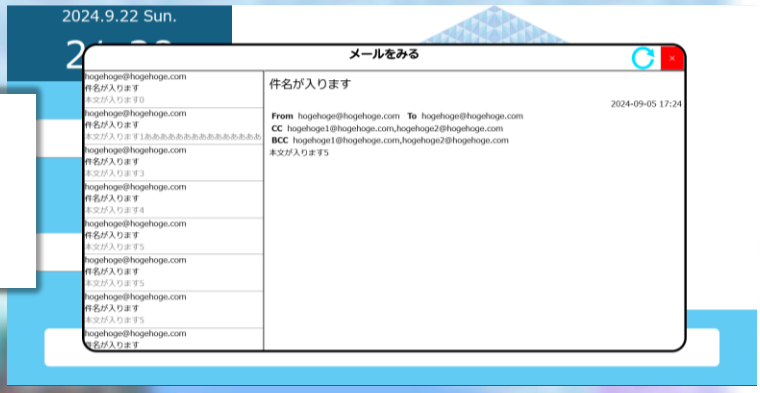
```
if(Distance[i] < 100){
    btfMove(Butterfly[i],Angle[i] + 180,EscapeSpeed);
    MoveTimer[i] = 0;
}
```

```
else{
    if(MoveTimer[i] > 0){
        btfMove(Butterfly[i],MoveAngle[i],MoveSpeed);
        MoveTimer[i] -= 1;
    }else{
        MoveAngle[i] = Math.floor(Math.random() * 1000 % 360);
        MoveTimer[i] = Math.floor(Math.random() * 1000 % 50);
    }
}
```

```
//移動
function btfMove(obj,d,speed){
    var i = Math.cos(Math.PI / 180 * d);
    var j = Math.sin(Math.PI / 180 * d);
    obj.x += i * speed;
    obj.y -= j * speed;
}
```

どちらも最後にbtfMoveという関数に、角度と移動速度を引数で渡し、移動を行います。

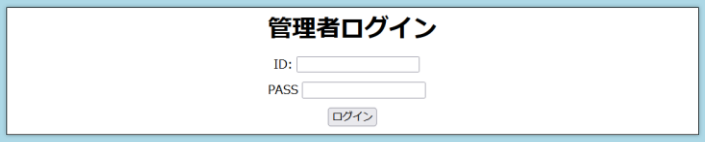
YUKIPOST

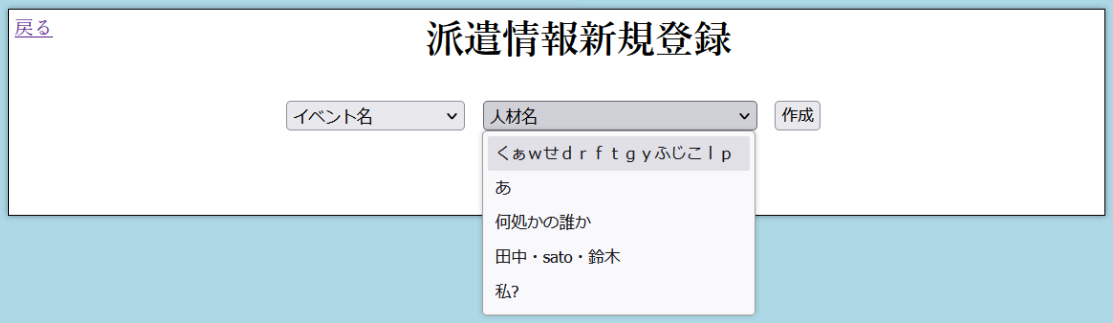


ジャンル：メールビューア
制作期間：２カ月
担当箇所：フロントエンド

こちらは、中島由貴オフィシャルファンクラブにて公開される作品で、私はフロントエンドを担当しています。
https://x.com/NYuki_OFC/status/1837750686467166536
ここでは、メールの送信、閲覧をすることができ、それに応じて部屋にいるキャラクターがアクションを起こしたりします。

人材管理システム





これは、管理者がどのイベントにどの人材を派遣するかを管理するシステムです。
イベント・人材情報を新規登録、編集、削除でき、その二つの情報を使って派遣情報を作成できます。

PHPのフレームワークであるLaravelを使用して制作していて、
データはデータベースに保存されるようになっています。

新しいイベント、人材、派遣情報を登録するときは、
バリデーションし必要なステータスがあるか確認し、
パスワードはハッシュ化してから保存しています。

```php
public function index()
{
    $workers = Worker::all();
    return view('admin.worker.index',['workers'=>$workers]);
}

public function create()
{
    return view('admin.worker.create');
}

public function store(Request $request)
{
    $validator = Validator::make($request->all(),
        [
            'name'=>'required',
            'email'=>'required',
            'password'=>'required',
            'memo'=>'max:255'
        ]
    );
    if($validator->fails()){
        return redirect('/admin/worker')->withErrors(["エラーが発生しました"]);
    }
    $validated = $validator->validated();
    $worker = new Worker;
    $worker->name = $validated['name'];
    $worker->email = $validated['email'];
    $worker->password = Hash::make($validated['password']);
    $worker->memo = $validated['memo'];
    $worker->save();
    return redirect('/admin/worker');
}
```

```blade
<div class="wrap">
<a href="/admin/menu" class="back">戻る</a>
<h1>派遣情報</h1>
<a href="/admin/dispatch/create">派遣情報新規登録</a>
<table border="1" class="table">
    <tr>
        <th>イベント名</th>
        <th>人材名</th>
        <th></th>
        <th></th>
    </tr>
    @foreach($dispatches as $dispatch)
    <tr>
        <td>{{$eventId[$count]}}</td>
        <td>{{$workerId[$count]}}</td>
        <td><a href="/admin/dispatch/{{$dispatch->id}}/edit">編集</a></td>
        <td>
            <form method="post" action="/admin/dispatch/{{$dispatch->id}}">
                @csrf
                @method('delete')
                <input type="submit" value="削除">
            </form>
        </td>

    </tr>
    <?php $count++; ?>
    @endforeach
</table>
</div>
```

```php
$eventId = array(null);
foreach($dispatches as $dispatch){
    foreach($events as $event){
        if($dispatch->event_id == $event->id){
            array_push($eventId,$event->title);
            break;
        }
    }
}
```

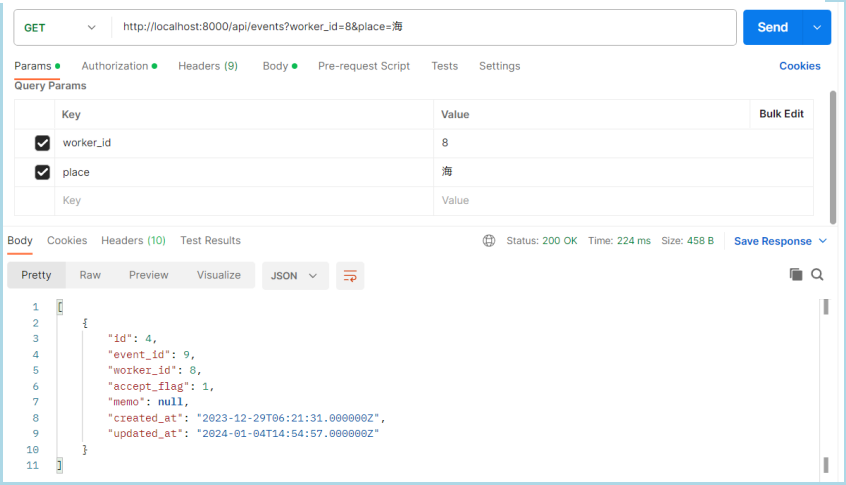
派遣情報はイベントのidと人材のidを格納しています。
派遣情報一覧では、イベント名と人材名を表示するので、
データを取ってきて必要なイベント名と人材名を配列に入れて、テーブルに出力しています。

API

人材のidとイベントの場所や日時を指定すると、その条件に合う派遣情報を検索できるAPIと、①

承認したい派遣情報のイベントidと人材idを指定すると、その派遣情報の承認フラグをtrueにできるAPIを作成しました。②

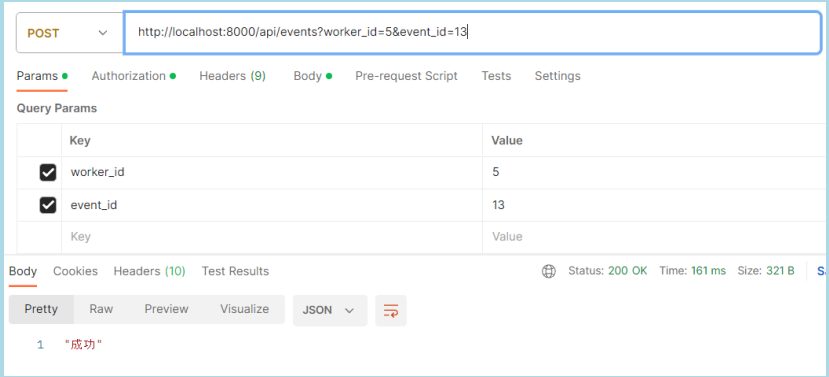
①



派遣情報のテーブル　　↓承認フラグ

	id	event_id	worker_id	accept_flag	memo	created_at	updated_at
編集 コピー 削除	4	9	8	1	NULL	2023-12-29 15:21:31	2024-01-04 23:54:57
編集 コピー 削除	5	8	7	0	NULL	2023-12-29 23:06:47	2024-01-04 23:54:47
編集 コピー 削除	6	10	9	0	NULL	2023-12-29 23:06:52	2024-01-04 23:55:03
編集 コピー 削除	7	13	5	1	NULL	2023-12-30 00:47:10	2024-01-05 00:47:55

②




```php
public function acceptEventData(Request $request){
    if(!($request->worker_id||$request->event_id)){
        return response()->json(['エラー'],404);
    }
    $data = Dispatch::where('worker_id',$request->worker_id)->where('event_id',$request->event_id)->get();
    if(empty($data[0]->id)){
        return response()->json('エラー',404);
    }
    for($i = 0;$i < count($data); $i++){
        $data[$i]->accept_flag = true;
        $data[$i]->save();
    }
    return response()->json('成功',200);
}
```

このゲームは箱を押して道を作りゴールを目指す、ブラウザ上で動作するゲームです。

この制作にはVueというフレームワークを利用しています。Vueの機能のv-ifを使い現在のステータスに応じて、表示するhtmlのタグを制御することで、ページ遷移をせずすべての画面を表示しています。①
ログインやフィールド情報の取得、リザルトにはAPIとの通信を行い情報を取得しています。②
プレイヤーと箱の移動は同じmoveという関数で動かしています。自分のナンバーと移動先のナンバーから移動できるか否かを判別し処理しています。③

```
<div class="login" v-if="status=='login'">…
</div>

<div class="select" v-if="status=='select'">…
</div>

<div class="profile" v-if="status=='profile'">…
</div>

<div class="game" v-if="status=='game'">…
</div>

<div class="result" v-if="status=='result'">…
</div>
```

①

```
// 認証成功
if ( $headers["Authorization"] == "Bearer " . $token ) {
    $responseCode = 200;
        $data = [
            "objects" => [
                [
                    0,0,1,1,1,0,0
                ],
                [
                    0,1,1,0,1,1,0
                ],
                [
                    1,1,0,3,0,1,1
                ],
                [
                    1,2,3,0,0,4,1
                ],
                [
                    1,1,0,3,0,1,1
                ],
                [
                    0,1,1,0,1,1,0
                ],
                [
                    0,0,1,1,1,0,0
                ]
            ]
        ];
```

②

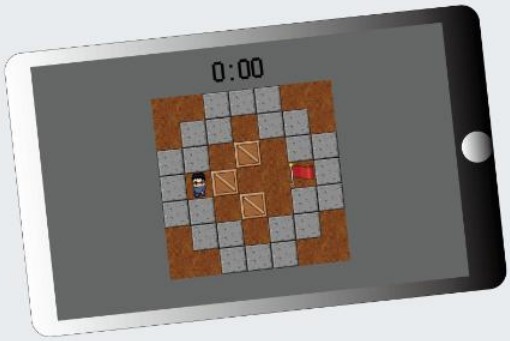
```
var fieldNum = this.gameInfo.fieldData[moveTo.y][moveTo.x];
var targetFieldNum = this.gameInfo.fieldData[target.y][target.x];

console.log(targetFieldNum + "が" + fieldNum + "に移動しようとしています。");

if(fieldNum == 0){
    this.gameInfo.fieldData[moveTo.y][moveTo.x] = targetFieldNum;
    this.gameInfo.fieldData[target.y][target.x] = fieldNum;

    if(targetFieldNum == 2){
        this.gameInfo.playerPos.x = moveTo.x;
        this.gameInfo.playerPos.y = moveTo.y;

        this.gameInfo.moveCount++;
    }

    return true;
} else if(fieldNum == 1){
    return false;
} else if(fieldNum == 3){
    if(targetFieldNum != 3){
        if(this.move(moveTo,key)){
            this.move(target,key);
            return true;
        }
        else{
            return false;
        }
    }
    else{
        return false;
    }
} else if(fieldNum == 4){
    if(targetFieldNum == 2){
        this.gameInfo.moveCount++;

        this.gameClear();
```

③

このゲームを宣伝するホームページも作成しました。
ここではアプリとして公開した想定で作っています。

Box Adventure

知恵、勇気、スピード



「Box Adventure」はPC、スマートフォンそれぞれのアプリストアからダウンロードし、インストールすることで遊ぶことが出来ます。

ダウンロード

STORY

勇者が閉じ込められたのは魔法の箱と謎が満ちる空間。剣を失ったいま試されるのは知恵と勇気、そしてスピードである。無尽蔵に増え続けるフロアを走り抜け、謎を解き明かせ……！

Event

・マンスリータイムアタック開催！

12月度のマンスリータイムアタックが間もなく開催されます！12/1～14に出したタイムの上位入賞者はランキングの名前の色が変化します。

News

・【メンテナンス】11/30 定期メンテナンス

以下の日時においてメンテナンスを実施いたします。 11/30(木) 14:00 ～ 15:00 メンテナンス中のログインはお控えください。 ユーザーの皆様にはご迷惑をおかけしますが、ご協力よろしくお願いいたします。

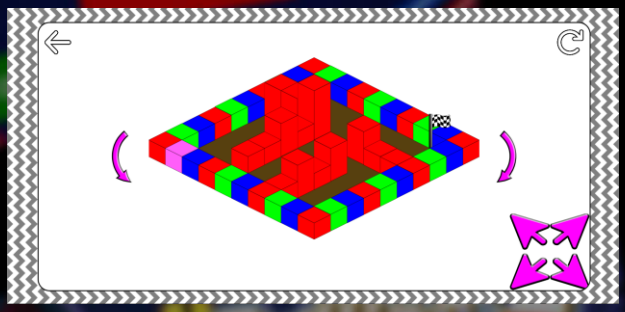
・【重要】対応端末変更のお知らせ

2023年12月以降のアップデートにて、動作保証の端末が変更されます。 対象端末につきましては、後日あらためてお知らせいたしますのでご了承ください。

その他制作物

TRICKMAZE
トリックメイズ

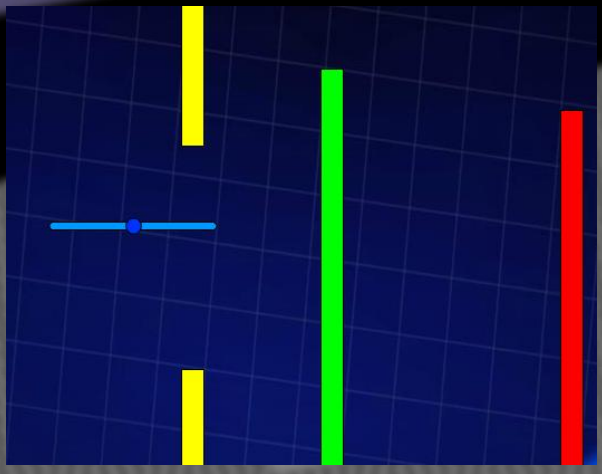
ジャンル：パズル
制作期間：1週間
言語：JavaScript、HTML/CSS



ピンクの箱を操作してゴールへ導く、トリックアートのような迷路のゲームです。
２Dのイラストを組み合わせて３Dのように見えるように作っています。

ランダム生成イライラ棒

ジャンル：イライラ棒
制作期間：１週間
言語：ActionScript3.0



Adobe Animateで制作したイライラ棒ゲームです。
PC用とAndroid用の2パターンがあります。