総合学園ヒューマンアカデミー

横浜校

ゲームカレッジ

プログラマー専攻

角屋潤

PORTFOLIO

ポートフォリオ

目次

・プロフィール

氏名：角屋 潤
性別：男
年齢：19歳
希望職種：プログラマー

所持スキル

言語

開発ツール

開発支援ツール

・アピールポイント

私はチーム制作でリードプログラマーを、２度務めた経験があります。

また、技能五輪のウェブデザイン職種に出場した経験があり、ウェブやサーバーサイドに関する知識も持っています。

制作作品

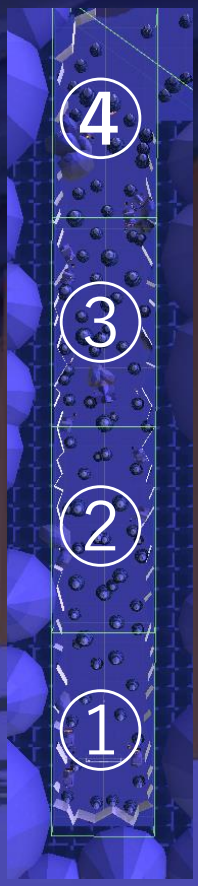
このゲームは、雪だるまを転がして大きくしながら
ゴールである家を目指す、アクションゲームです。

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。

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が、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。

あり	なし
	

CSVデータを読み込みマップを生成するEditor拡張

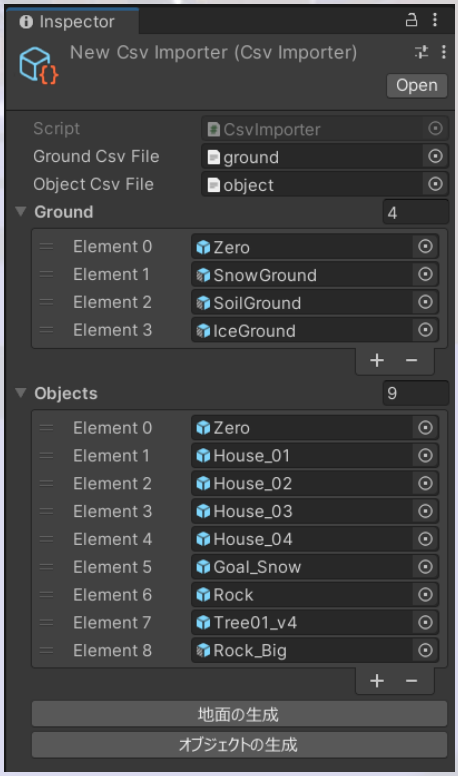
数値を入れたCSVデータを読み込み、対応した地面やオブジェクトを配置できるEditor拡張を作成しました。

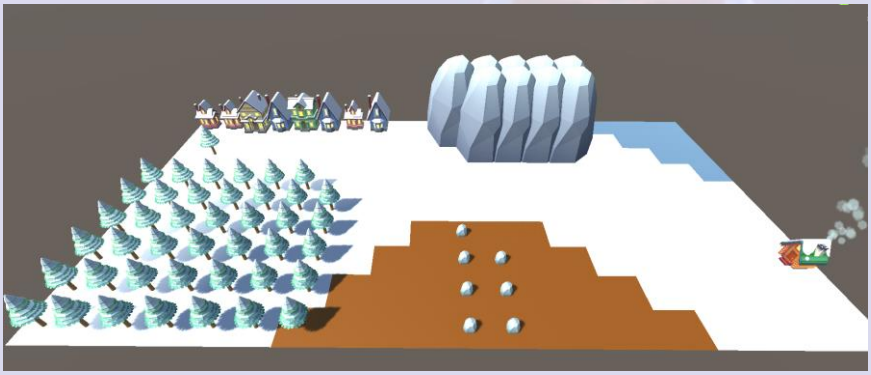
ground.csv

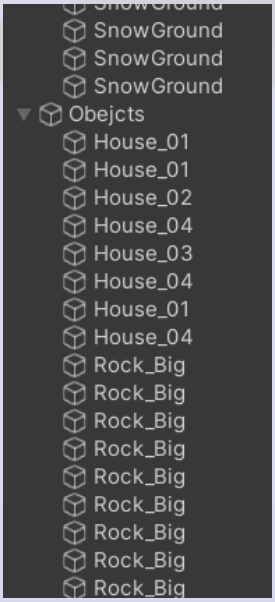
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3	3	3	3
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3	3	3
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3	3
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1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

このゲームの製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。ほかにも製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。

魚の操作

```
void Update()
{
    //角度に応じてジャンプパワーのベクトルを変更する
    float rad = Mathf.Deg2Rad * transform.eulerAngles.z;
    float sinvalue = Mathf.Sin(rad);
    float cosvalue = Mathf.Cos(rad);
    Vector2 vector2 = new Vector2(cosvalue,sinvalue);
    jumpVector = vector2 * jumpPower;

    //縦に近いほど減速
    if(rb.velocity.x > 0)
    {
        Vector3 velocity = rb.velocity;
        velocity.x -= Mathf.Abs(sinvalue) * Time.deltaTime * downSpeed;
        rb.velocity = velocity;
    }

    //上昇中y方向の速度を下げる
    if(rb.velocity.y > 0)
    {
        Vector3 upVelocity = rb.velocity;
        upVelocity.y -= upDownSpeed * Time.deltaTime;
        rb.velocity = upVelocity;
    }
```

①

②

③

魚の操作についてはまず、プレイヤーの操作で回転ができ向いている方向にジャンプができるようにする、とのことだったので、そのベクトルをeulerAngles.zの値をsin,cosに変換することで求めました。（①）
竜巻のギミックもこの仕組みを利用しています。
そして角度によって速度を変える仕様もsinの絶対値を利用して実装しています。（②）
魚の落下速度は遅くするため魚にかかる重力を軽くしました。その結果上を向いてジャンプした場合、想定以上に上に飛んでしまったので、上昇中のみ下向きの力を加えることで対処しました。（③）

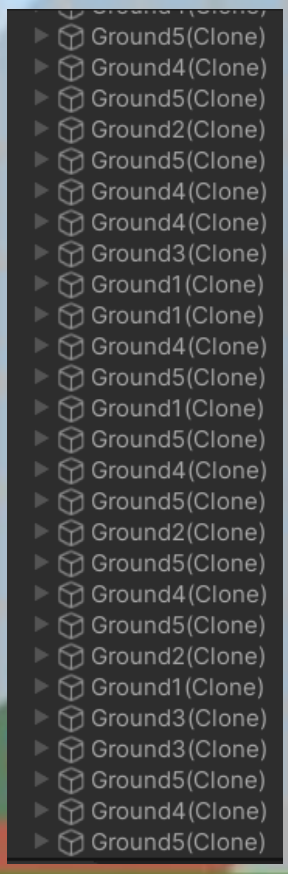
竜巻

```
if (tornadeNo == 2 || tornadeNo == 3)
{
    //オブジェクトの移動
    MoveCycle += Time.deltaTime * tornadeCycle;
    float MovePos = Mathf.Sin(MoveCycle * Mathf.Deg2Rad);
    transform.position += new Vector3((MovePos * tornadeMove.x * Time.deltaTime), (MovePos * tornadeMove.y * Time.deltaTime), 0);
    if (MoveCycle > 360)
    {
        MoveCycle = 0;
    }
}
```

竜巻は触れたオブジェクトを竜巻の向きに
吹き飛ばすオブジェクトです。
この竜巻には、周期的に移動するように
することができます。
これには１から-1の値をとるsinを活用し
実装しています。
（現段階ではこの移動する竜巻は使用されていません）

ステージのランダム生成

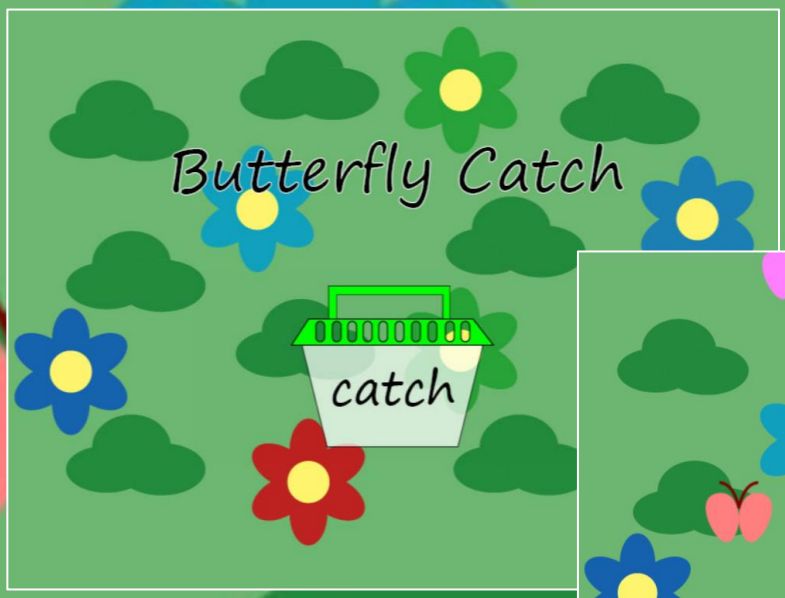
```
private void Start()
{
    pauseUI.SetActive(false);
    for (int i = 0; i < 30; i++)
    {
        int groundNumber = UnityEngine.Random.Range(0, ground.Length);
        Vector3 setGroundPos = new Vector3(200 + 90 * i, -0.5f, 0);
        Instantiate(ground[groundNumber], setGroundPos, Quaternion.identity);
    }
}
Unity メッセージ | 0 個の参照
```

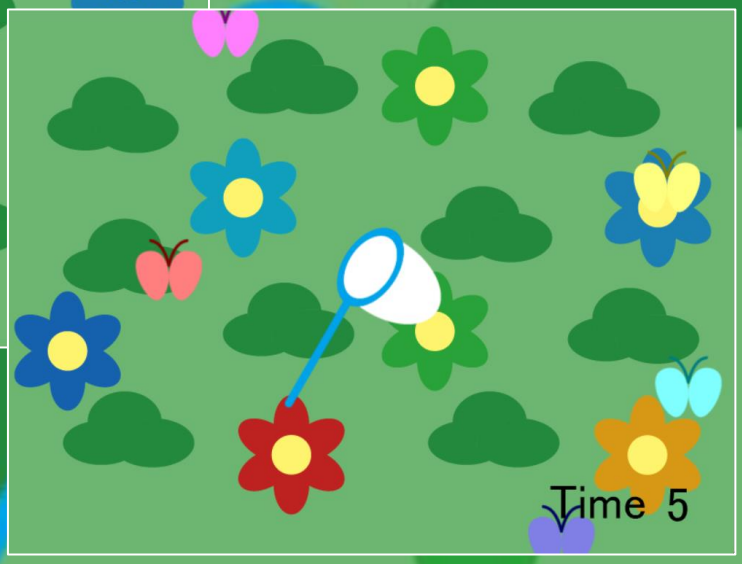


ステージはPrefab化された横９０ｍのパーツ
を５つ用意し、それをスタート時にランダム
で３０個生成しています。
生成する座標に２００を足しているのは、
最初に何もない地面が２００ｍあるからです。

Butterfly Catch

ジャンル：虫取りゲーム
制作人数：1人
製作期間：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。

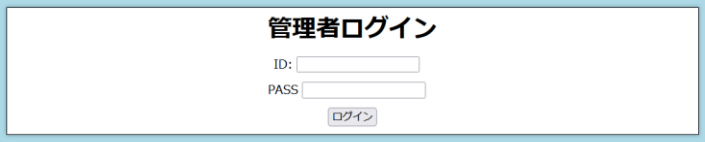
```
if(Distance[i] < 100){
    btfMove(Butterfly[i],Angle[i] + 180,EscapeSpeed);
    MoveTimer[i] = 0;
}
```

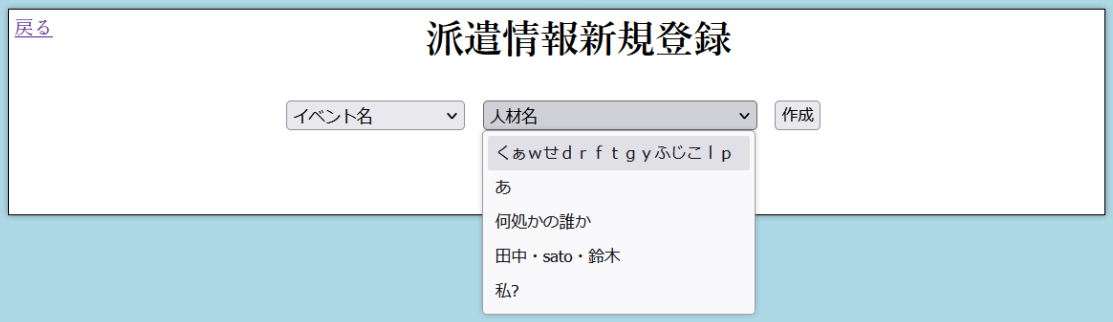
```
else{
    if(MoveTimer[i] > 0){
        btfMove(Butterfly[i],MoveAngle[i],MoveSpeed);
        MoveTimer[i] -= 1;
    }else{
        MoveAngle[i] = Math.floor(Math.random() * 1000 % 360);
        MoveTimer[i] = Math.floor(Math.random() * 1000 % 50);
    }
}
```

```
//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。

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;
        }
    }
}
```

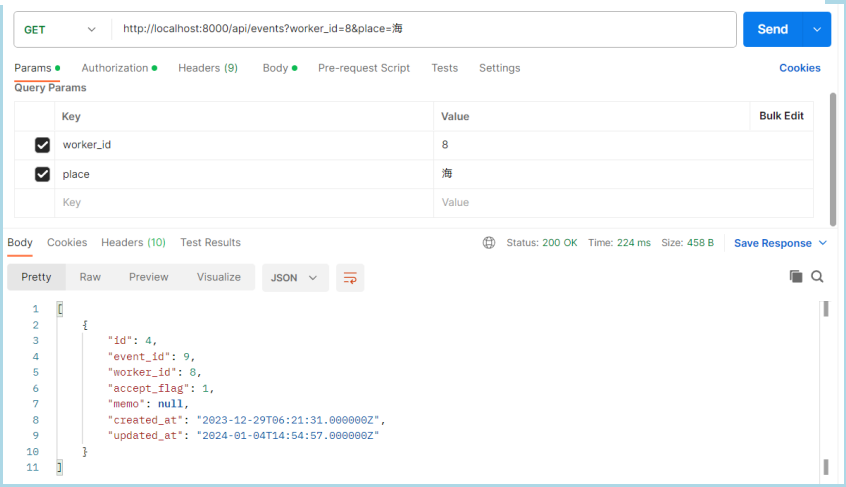
派遣情報はイベントのidと人材のidを格納しています。
派遣情報一覧では、イベント名と人材名を表示するので、
データを取ってきて必要なイベント名と人材名を配列に入れて、テーブルに出力しています。

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

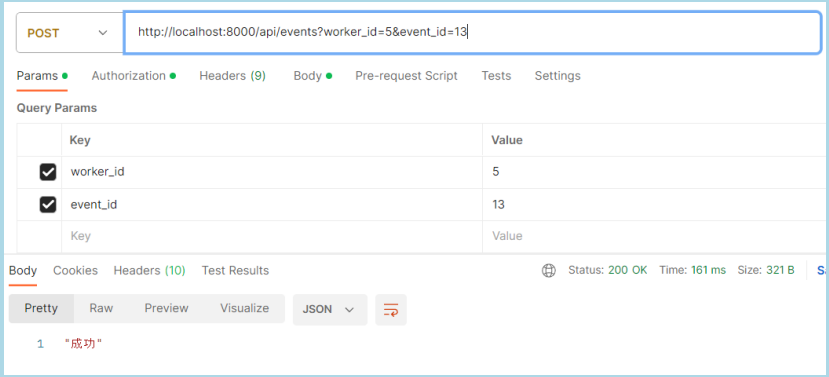
①



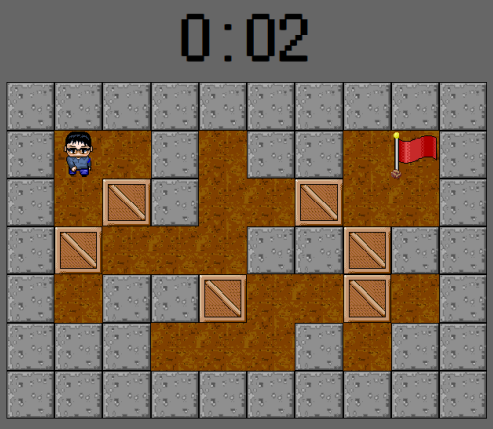
派遣情報のテーブル　　↓承認フラグ

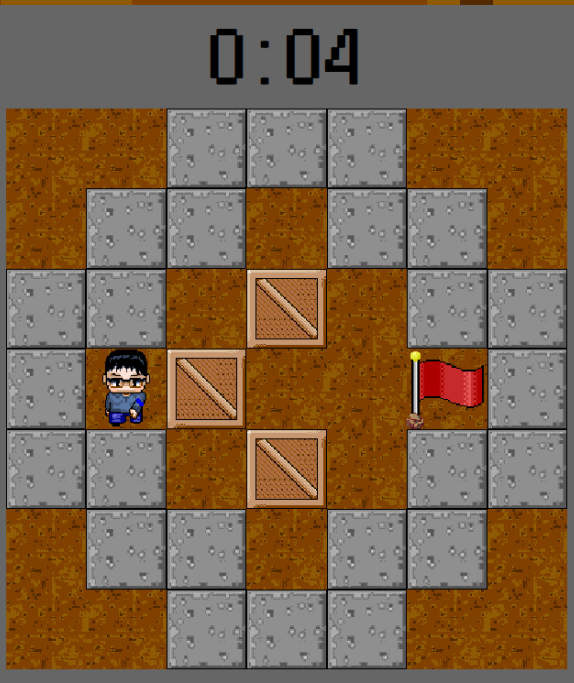
	id	event_id	worker_id	accept_flag	memo	created_at	updated_at
編集 コピー 削除	4	9	8	1	NULL	2023-12-29 15:21:31	2024-01-04 23:54:57
編集 コピー 削除	5	8	7	0	NULL	2023-12-29 23:06:47	2024-01-04 23:54:47
編集 コピー 削除	6	10	9	0	NULL	2023-12-29 23:06:52	2024-01-04 23:55:03
編集 コピー 削除	7	13	5	1	NULL	2023-12-30 00:47:10	2024-01-05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

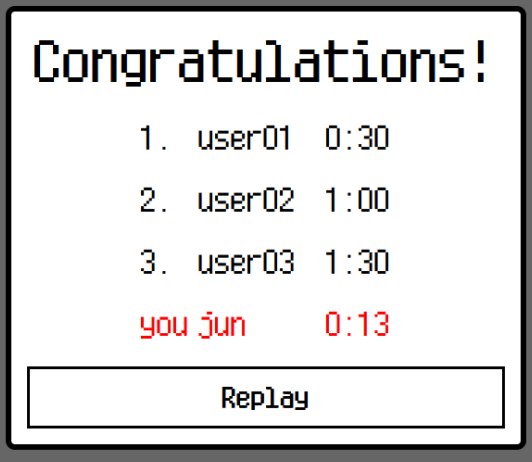
②




```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

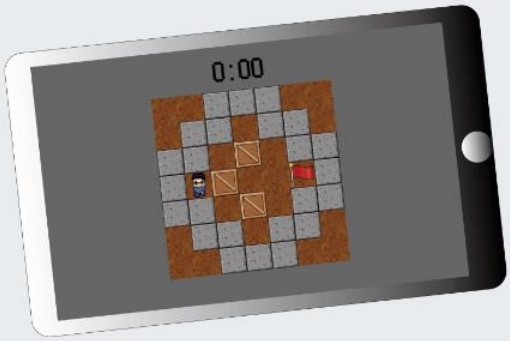
```
var fieldNum = this.gameInfo.fieldData[moveTo.y][moveTo.x];
var targetFieldNum = this.gameInfo.fieldData[target.y][target.x];

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

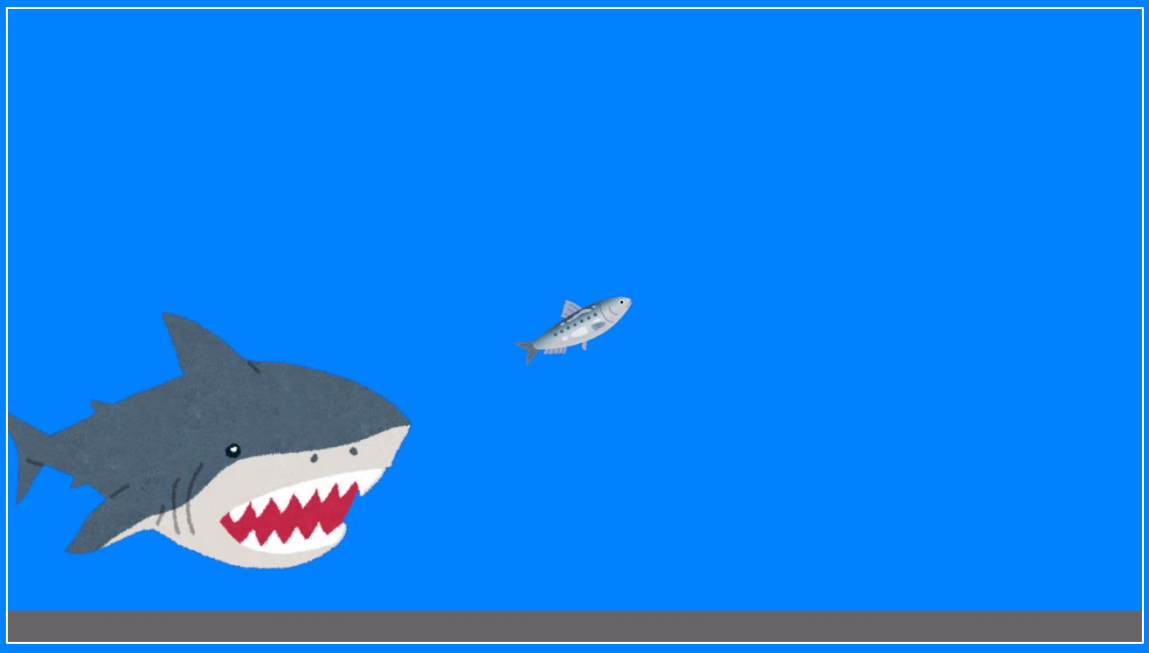
以下の日時においてメンテナンスを実施いたします。 11/30(木) 14:00 ～ 15:00 メンテナンス中のログインはお控えください。 ユーザーの皆様にはご迷惑をおかけしますが、ご協力よろしくお願いいたします。

・【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。

FishEscape

ジャンル：２Ｄアクション
制作人数：１人
制作期間：１週間
このゲームは魚を操作して、サメから逃げてゴールを目指すゲームです。
サメはプレイヤーの魚を常に追いかけてきます。



サメのコントロール

サメとプレイヤーの位置との差を出し、ｘの方が大きければ
ｙの値をｘで割りｘはｘの絶対値で割る。
ｙの方が大きければその逆をして、値を１以下にし、
サメの移動に使えるベクトルにしました。
そして向きが変わらないのは変なので、
Atan2を使い、ｘとｙの値から角度をもとめ回転させました。

```
//span秒ごとにplayerの向きに力を加える
delta += Time.deltaTime;
if (delta > span)
{
    //プレイヤーのいる方向を求め、その方向に力を加える
    Vector2 vector3 = new Vector3(player.transform.position.x - transform.position.x, player.transform.position.y - transform.position.y, 0);
    if (Mathf.Abs(vector3.x) > Mathf.Abs(vector3.y))
    {
        vector3.y /= vector3.x;
        vector3.x /= Mathf.Abs(vector3.x);
    }
    else
    {
        vector3.x /= vector3.y;
        vector3.y /= Mathf.Abs(vector3.y);
    }
    rb.AddForce(vector3 * dashPower, ForceMode2D.Impulse);
    //力を加えた向きに、オブジェクトを回転させる
    Vector3 SharkRot = new Vector3(0, 0, Mathf.Atan2(vector3.y, vector3.x) * Mathf.Rad2Deg);
    transform.localEulerAngles = SharkRot;
    delta = 0;
}
```

カメラ

カメラはプレイヤーを追尾して映しますが、ステージには
それ以上進めない場所があるので、clampを使いカメラが
進めない場所を映すことがないようになっています。

```
[SerializeField]
[Tooltip("追従するオブジェクト")]
private GameObject fish = null;

[SerializeField]
[Tooltip("魚の画面中央からのズレ")]
private Vector3 cameraFish = Vector3.zero;

float MAX = 10;
float MIN = 0;
参Unity メッセージ|0 個の参照
void Update()
{
    Vector3 cameraPos = fish.transform.position;
    cameraPos.x = Mathf.Clamp(cameraPos.x, 0, Mathf.Infinity) + cameraFish.x; //cameraFish分だけずらす
    cameraPos.y = Mathf.Clamp(cameraPos.y, MIN, MAX) + cameraFish.y; //cameraFish分だけずらす
    transform.position = cameraPos;
}
```

ランダム生成イライラ棒

ジャンル：イライラ棒
制作人数：1人
製作期間：１週間