매 시간 발전하는 개발자

김웅섭

소개

프로필

이름 : 김웅섭

생년월일 : 2006. 03. 27

전화번호 : 010-4060-1993

이력

2023 지방 기능 경기대회 출전
2024 지방 기능 경기대회 출전(5위)
2022 교내 여름방학 게임잼 출전
2023 교내 여름방학 게임잼 출전
2024 교과 우수상(게임 프로그래밍)
GTQ Excel 자격증

목차

1. 디지텍 컴퍼니
2. 슬라임 디펜스
3. 2023 전국 기능대회 연습작
4. 2024 지방 기능대회 연습작
5. 몬스터 스크램블
6. 핀볼 좀비
7. Escape From
8. 길찾기 알고리즘 연구

디지텍 컴퍼니

링크 :

https://github.com/miro03...

플랫폼 : PC

사용 언어 : C#

개발 인원 : 프로그래밍 2

아트 2

기획 1

개발 기간 : 1개월

역할 : 메인 프로그래밍

#리썰 컴퍼니 #졸업 작품

```
private void DoInteract()
{
    if (!photonView.IsMine) return;
    if (Physics.Raycast(cam.transform.position, cam.transform.forward, out var hit, interactionDistance, ~LayerMask.GetMask("Player")))
        hit.collider.TryGetComponent(out lookInteractable);
    else
        lookInteractable = null;

    if (lookInteractable != null && LookInteractable.IsInteractable(this))
    {
        var targetInteractId = lookInteractable.GetTargetInteractID(this);
        var interactAction = GetInteractInputAction(targetInteractId);

        //if pressed target interact id -> mark interact is start
        if (interactAction.WasPressedThisFrame())
            interactRequireTimes[(int)lookInteractable.GetTargetInteractID(this)] += Time.deltaTime;

        //if released target interact id -> set require time 0
        if (interactAction.WasReleasedThisFrame())
            interactRequireTimes[(int)lookInteractable.GetTargetInteractID(this)] = 0;

        //if interact marked
        if (interactRequireTimes[(int)lookInteractable.GetTargetInteractID(this)] > 0)
        {
            //less than require
            if (interactRequireTimes[(int)lookInteractable.GetTargetInteractID(this)] < lookInteractable.GetInteractRequireTime(this))
            {
                interactRequireTimes[(int)lookInteractable.GetTargetInteractID(this)] += Time.deltaTime;
            }
            else //reached
            {
                interactRequireTimes[(int)lookInteractable.GetTargetInteractID(this)] = 0;

                if (lookInteractable is ItemBase)
                {
                    var item = lookInteractable as ItemBase;
                    itemContainer.InsertItem(item);
                    item.OnInteract(this);
                    item.OnActive();
                }
                else
                {
                    lookInteractable.OnInteract(this);
                }
```

● 게임 설명

교내 졸업 작품에서 발표하기 위하여
만드는 프로젝트.

리썰 컴퍼니의 아이템과 적을 디지텍고의 여러 재미있는 상황들과 물건들로 교체하면 재미있을 것 같아서 만들었다.

플레이어, 아이템, 전체적인 게임 플로우, 서버, 게임 구조 구축을 담당하였다.

● 활용 기술

- Unity Photon을 사용하여 서버를 구축하였다.

- 기존 게임에서 지원하지 않던 중도 입장을 구현하기 위하여 Request → Receive 형태의 흐름을 설계하였다.

- 네트워크 연산의 최적화를 위하여 Observer Pattern을 적용시켰다.

슬라임 디펜스

링크 :

https://github.com/Proffe..

플랫폼 : 모바일

사용 언어 : C#

개발 인원 : 프로그래밍 2

기획 1

개발 기간 : 1개월

역할 : 메인 프로그래밍

#출시 목적 #디펜스

#슬라임 #모바일


```
/// <param name="base">base stats</param>
public void Calculate(Key key, Stats target, Stats @base)
{
    target.SetStat(key, x => @base.GetStat(key) * (percentValues.GetStat(key) + 1));
    target.SetStat(key, x => x + addValues.GetStat(key));
}

public override string ToString()
{
    var sb = new StringBuilder();
    sb.Append("percent\n");
    sb.Append(percentValues.ToString()).Append("\n");
    sb.Append("add\n");
    sb.Append(addValues.ToString());
    return sb.ToString();
}

public string Save()
{
    var casterData = new StringListWrapper();
    foreach (var c in casterInfo)
    {
        var infoData = new StringListWrapper();
        foreach (var i in c.Value as IDictionary<Key, Info>)
            infoData.datas.Add($"{i.Key}\'{i.Value.add.Value},{i.Value.percent.Value}");
        casterData.datas.Add($"{c.Key}\'{JsonUtility.ToJson(infoData)}");
    }
    return JsonUtility.ToJson(casterData);
}

public void Load(string data)
{
    if (string.IsNullOrEmpty(data)) return;
    var casterData = JsonUtility.FromJson<StringListWrapper>(data);
    foreach (var c in casterData.datas)
    {
        var caster = c[0..c.IndexOf('\'')];
        var infoJson = c[(c.IndexOf('\'') + 1)..];
        var infoData = JsonUtility.FromJson<StringListWrapper>(infoJson);
        foreach (var i in infoData.datas)
        {
            var key = Enum.Parse<Key>(i[0..i.IndexOf('\'')]);
            var values = i[(i.IndexOf('\'') + 1)..];
            var addValue = float.Parse(values.Split(',')[0]);
            var percentValue = float.Parse(values.Split(',')[1]);
            Set
            (
                caster,
                key,
                x => percentValue,
                x => addValue
            );
        }
    }
}
```

● 게임 설명

구글 플레이 스토어에
출시를 목적으로 만들던
디펜스 게임.

모든 라운드 클리어 또는 최대한 오래 살아남는 것이 목표이다.

아군 슬라임은 같은 종류와 같은 레벨인 다른 슬라임과 합치면 된다.
이외에도 증강체, 구조물 파괴 등 전략적으로 스테이지를 헤쳐나가며 클리어 해야되는 게임이다.

인게임의 모든 코드를 담당하였다.

● 활용 기술

- Google Sheet에서 데이터를 자동으로 파싱하여 사용

- Observer pattern를 사용해 스탯 시스템 구현

- Batching, Occlusion culling 등 사용하여 최적화

2023 전국 기능대회 연습

링크 :

https://github.com/Proffe..

플랫폼 : PC

사용 언어 : C#

개발 인원 : 프로그래밍 1

개발 기간 : 3일

역할 : 메인 프로그래밍

#2023 기능경기대회

#디펜스 게임 #2.5D

● 게임 설명

2023 전국기능경기대회
에서 만들기 위하여
제작하였던 타워 디펜스 프로젝트

본선에 나가는 친구를 도와주는 도우미로서 기능경기대회를 간접적으로 참여하였다.

```
using UnityEngine;
using UnityEngine.AI;

public class DisappearRecon : Recon, ISelectable
{
    private int remainTime = 60;

    public override string ExplainContent =>
        $"Remain. {remainTime / 60:0}:{remainTime:00}\n" +
        $"DPS. {1 / stats[0].attackDelay * stats[0].damage:0.##}\n" +
        $"Speed. {stats[0].speed}\n\n" +
        explain;

    public override int RequireCost => stats[0].nextRequireCost;

    public void Init()
    {
        var gm = Singleton.Get<GameManager>();
        var rand50p = Random.Range(0, 2) == 0;
        if (gm.ExistSubCastles && rand50p)
        {
            var randangle = Random.Range(0, 360);
            var pos =
                new Vector3(Mathf.Cos(randangle * Mathf.Deg2Rad), 0, Mathf.Sin(randangle * Mathf.Deg2Rad)) * 1.75f +
                gm.SubCastles[Random.Range(0, gm.SubCastles.Length)].transform.position;
            pos.y = 10;
            transform.position = pos;
            Debug.Log(pos);
        }
        else
        {
            var randangle = Random.Range(0, 360);
            var pos =
                new Vector3(Mathf.Cos(randangle * Mathf.Deg2Rad), 0, Mathf.Sin(randangle * Mathf.Deg2Rad)) * 1.75f +
                gm.MainCastle.transform.position;
            pos.y = 10;
            transform.position = pos;
            Debug.Log(pos);
        }

        StartCoroutine(MoveRoutine());
        StartCoroutine(DisappearRoutine());
    }

    protected override void Start()
    {
        agent = GetComponent<NavMeshAgent>();
        recons.Add(this);
    }

    private IEnumerator DisappearRoutine()
    {
        while(remainTime-- > 0) yield return new WaitForSeconds(1f);
        recons.Remove(this);
```

● 활용 기술

- 2.5D 구현을 위해 캐릭터에 빌보드 시스템을 적용

- Static class extension, Lerp를 이용해 간단하게 Tween System 구현

2024 지방 기능대회 연습

링크 :

https://github.com/Proffe..

플랫폼 : PC

사용 언어 : C#

개발 인원 : 프로그래밍 1

개발 기간 : 1주일

역할 : 메인 프로그래밍

#지방기능 경기대회 #레이싱 게임


```
public class Player : BaseFlighter
{
    public static Player Instance { get; private set; }

    private DataManager dataManager => DataManager.Instance;
    private GameManager gameManager => GameManager.Instance;
    private ResourceContainer resourceContainer => ResourceContainer.Instance;

    public Transform orientation;
    public LayerMask modelAlignLayer;

    [NonSerialized] public int money;
    [NonSerialized] public float curMaxSpeed;
    [NonSerialized] public float curSteerSpeed;
    [NonSerialized] public new Rigidbody rigidbody;
    [NonSerialized] public AudioSource audioSource;
    [NonSerialized] public PlayerModel playerModel;

    public PlayerData PlayerData => dataManager.playerDatas[dataManager.playerSelect];
    public float Speed => rigidbody.velocity.magnitude;

    private void Awake()
    {
        Instance = this;
    }

    protected override void Start()
    {
        base.Start();
        rigidbody = GetComponent<Rigidbody>();
        audioSource = GetComponent<AudioSource>();

        playerModel = Instantiate(resourceContainer.playerModels[dataManager.playerSelect], orientation);
        playerModel.GetComponent<MeshRenderer>().sharedMaterial = resourceContainer.playerColors[dataManager.playerColorSelect];
        playerModel.transform.localPosition = Vector3.zero;
        model = playerModel.transform;

        curMaxSpeed = PlayerData.maxSpeed;
    }

    private void Update()
    {
        Physics.Raycast(model.position, Vector3.down, out var hit, 10f, modelAlignLayer);
        model.rotation = Quaternion.Lerp(model.rotation, Quaternion.FromToRotation(Vector3.up, hit.normal) * orientation.rotation, Time.deltaTime * 3);

        if (!GameManager.Instance.isGameRunning) return;
        if (!canMove) return;

        playerModel.isDrift = Input.GetKey(KeyCode.LeftShift);
        curSteerSpeed = Mathf.Lerp(curSteerSpeed, Input.GetKey(KeyCode.LeftShift) ? PlayerData.driftSteerSpeed : PlayerData.steerSpeed, Time.deltaTime * 2f);

        orientation.Rotate(0, Input.GetAxis("Horizontal") * curSteerSpeed * Time.deltaTime, 0);
```

● 게임 설명

2024 지방기능경기대회를 나가기 위하여 연습하였던 프로젝트

자동차를 업그레이드 하고, 장애물을 피하여 상대보다 먼저 결승선에 도달하면 되는 게임이다.

● 활용 기술

- Wheel Collider의 복잡함을 해소하기 위하여 Sphere Collider를 사용

- Static class extension, Lerp를 이용해 간단하게 Tween System 구현

몬스터 스크램블

링크 :

https://github.com/Proffe..

플랫폼 : PC

사용 언어 : C#

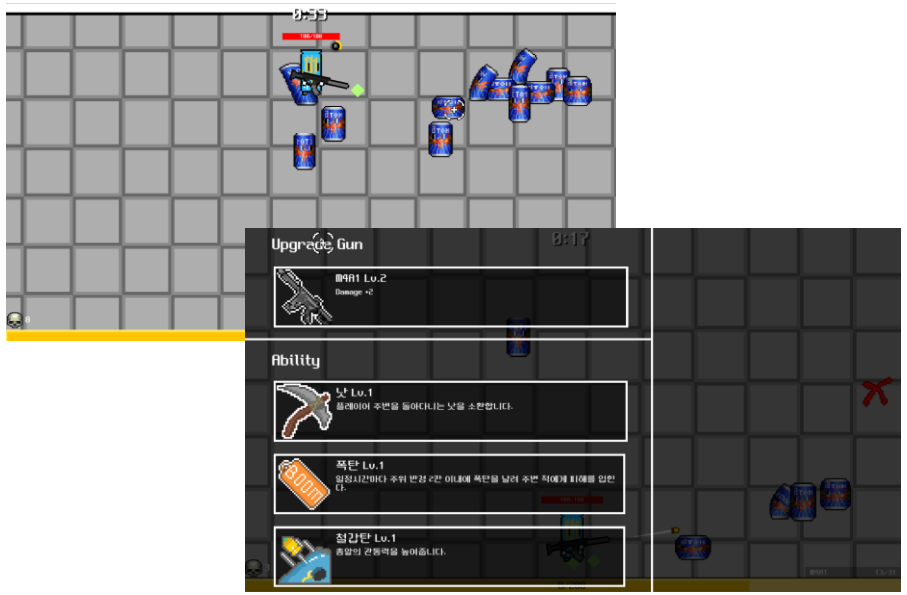
개발 인원 : 프로그래밍 1

그래픽 1

개발 기간 : 1개월

역할 : 메인 프로그래밍

#Brotato #뱀서라이크




```
    Instantiate(hitParticlePrefab, transform.position, Quaternion.identity);
}

private void Awake()
{
    SingletonManager.RegisterSingleton(this);
    anim = GetComponent<Animator>();
    rb = GetComponent<Rigidbody2D>();
    sr = GetComponent<SpriteRenderer>();

    curHp = maxHp;
    maxExp = 100;
    curLvl = 1;
    curBombWaitTime = bombWaitTime;
    UpgradeGun(startRifle);
    curGun.SetRotation(20);
    curGun.gameObject.SetActive(false);
    abilityLvls = new int[System.Enum.GetValues(typeof(AbilityType)).Length];
}

private void Update()
{
    if (abilitySelectUI.IsDisplayingUI) return;
    if (!gameManager.IsGameStart) return;

    if (CurHp <= 0)
    {
        curHp = 0;
        if (!isEnd)
        {
            isEnd = true;
            Time.timeScale = 0;
            backgroundSound.Fade(SoundFadeType.Out, 2.5f);
            this.InvokeRealTime(() => scoreUI.DisplayUI(), 3);
        }
        return;
    }

    //move
    var h = Input.GetAxisRaw("Horizontal");
    var v = Input.GetAxisRaw("Vertical");
    var isMove = h != 0 || v != 0;

    rb.velocity = new Vector2(h, v).normalized * moveSpeed;

    transform.position = new Vector3
    (
        Mathf.Clamp(transform.position.x, -14.45f, 14.45f),
        Mathf.Clamp(transform.position.y, -9.673f, 8.72f),
        0
    );
```

● 게임 설명

교내에서 프로그래머와
그래픽 디자이너를 Pair
로 짝지어서 진행하였던 프로젝트

Brotato 게임의 강화 능력을 레퍼런스
삼아서 만든 게임이다.

적을 잡아서 총과 능력을 업그레이드 해서
10분까지 버티면 되는 게임이다.

● 활용 기술

- 게임을 빠르게 완성하기 위하여
Scriptable Object와 Prefab
System을 적극 활용

- Singleton의 유연성과 결합도에 관련
된 문제를 해결하기 위하여
Service Locator 패턴을 사용

핀볼 좀비

링크 :

https://github.com/kim-jeo..

플랫폼 : PC

사용 언어 : C#

개발 인원 : 프로그래밍 1

그래픽 1

기획 1

개발 기간 : 3일

역할 : 메인 프로그래밍

#동아리 게임잼 #핀볼


```
    isWait = false;
}

int spawnPosX;
float spawnAmount = Random.Range(1, 5.5f);

int randomType;
for (int i = 0; i < spawnAmount; i++) // 좀비 랜덤 생성
{
    randomType = Random.Range(1, 100);
    spawnPosX = Random.Range(1, 5);
    if (objectInTile[spawnPosX] == null)
    {
        var enemy = ObjectPool.GetObject(ObjectPool.instance.prefebs[1], null);

        if (randomType <= 70)
            enemy.GetComponent<Enemy>().Enemytype = (int)EnemyType.Normal;
        else if (randomType > 70 && randomType <= 85)
            enemy.GetComponent<Enemy>().Enemytype = (int)EnemyType.Plague;
        else if (randomType > 85 && randomType <= 100)
            enemy.GetComponent<Enemy>().Enemytype = (int)EnemyType.Miner;

        enemy.transform.position = new Vector2(tilePos[spawnPosX].x, tilePos[spawnPosX].y + 3);
        enemy.GetComponent<SortingGroup>().sortingOrder = 2;
        enemy.transform.Find("Hp").GetComponent<SortingGroup>().sortingOrder = 4;
        enemy.transform.DOMove(tilePos[spawnPosX], 1.0f).SetEase(Ease.OutBounce);
        objectInTile[spawnPosX] = enemy;
    }
    yield return new WaitForSeconds(0.1f);
}

randomType = Random.Range(0, 2);
spawnPosX = Random.Range(0, 5);
if (objectInTile[spawnPosX] == null)
{
    GameObject Item;
    if (randomType == 0)
    {
        Item = ObjectPool.GetObject(ObjectPool.instance.prefebs[2], null);
        Item.GetComponent<Item>().item = (int)ItemType.heal;
    }
    else
    {
        Item = ObjectPool.GetObject(ObjectPool.instance.prefebs[3], null);
        Item.GetComponent<Item>().item = (int)ItemType.bullet;
    }

    objectInTile[spawnPosX] = Item;
    Item.transform.position = new Vector2(tilePos[spawnPosX].x, tilePos[spawnPosX].y + 3);
    Item.transform.DOMove(tilePos[spawnPosX], 1.0f).SetEase(Ease.OutBounce);
}

spawnPosX = Random.Range(0, 5);
if (turn % 5 == 0)
{
    if (objectInTile[spawnPosX] != null)
    {
        if (objectInTile[spawnPosX].TryGetComponent(out Enemy enemy))
            enemy.currentHp = 0;
        else if (objectInTile[spawnPosX].TryGetComponent(out Item item))
            ObjectPool.ReturnObject(item.gameObject);
    }
    var Box = Instantiate(box);
    Box.transform.position = new Vector2(tilePos[spawnPosX].x, tilePos[spawnPosX].y + 3);
```

● 게임 설명

교내에서 진행하였던
게임잼에서 제작한
프로젝트

계속해서 나오는 좀비들을 처치하고 아이템 박스를 부숴서 자신을 강화하며 최대한 오래 살아남는 게임

● 활용 기술

- Dictionary를 사용하여 좀비의 행동과 수명을 관리

- 최적화를 하기 위하여 Object Pool을 사용

Escape From

링크 :

https://github.com/Proffe..

플랫폼 : PC

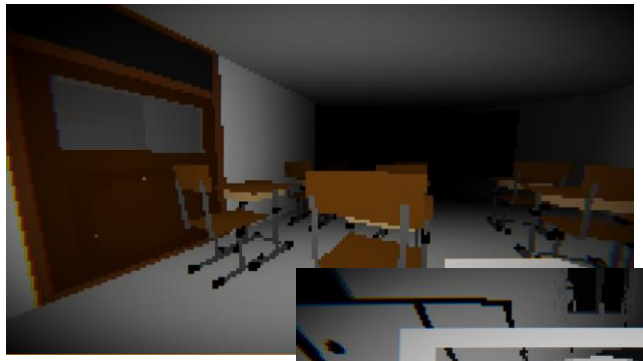
사용 언어 : C#

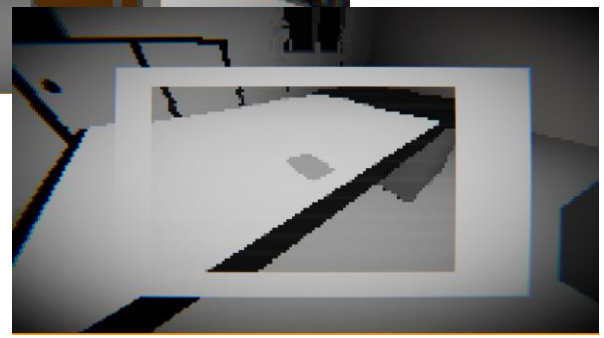
개발 인원 : 프로그래밍 2
그래픽 2
기획 1

개발 기간 : 1개월

역할 : 메인 프로그래밍

#공포 #Shader graph
#Pixelate





```
    transform.Rotate(new Vector3(0, mouseX * mouseSensivity.x * 100, 0));
}

private void Interact()
{
    RaycastHit hitInfo;
    Physics.Raycast(
        cameraHolder.transform.position,
        cameraHolder.transform.forward, out hitInfo,
        interactRayDistance,
        ~LayerMask.GetMask("Player", "Ignore Raycast")
        );

    if (hitInfo.collider != null)
    {
        IInteractable interactable;
        if (hitInfo.collider.TryGetComponent<IInteractable>(out interactable))
        {
            interactable.ShowUI();

            //criteria
            if (
                !EscUI.IsShowing &&
                !SettingUI.IsShowing &&
                InventoryUI.ShowType == InventoryUIShowType.disable &&
                StartMenuUI.IsStart &&
                !isDeath
            )
            {
                if (Input.GetKeyDown(KeyCode.E))
                {
                    interactable.Interact();
                }
            }
        }
        else InteractUI.ControlUI(false, "");
    }
    else InteractUI.ControlUI(false, "");
}

private void HeadHob()
{
    if (!headHobEnabled) return;

    CheckHeadHobMotion();
    ResetHeadPosition();
    _camera.LookAt(FocusTarget());
```

● 게임 설명

교내 동아리 게임잼에서
제작하였던 공포게임

각 층의 공략 방법 힌트를 사용자가 가진 카메라로 얻으면서 탈출하는게 목적이다.

IMSCARED라는 게임에서 영감을 받아 Pixelate 화면에 카메라로 아이템을 찍어서 얻으면 좋겠다는 생각으로 시도해본 프로젝트이다.

1층, 2층, 플레이어 부분을 담당하였다.

● 활용 기술

- 게임에 더욱 공포스러운 연출을 부각하기 위하여 Pixelate, Retro Shader Graph를 구현 및 활용

- URP로 진행한 프로젝트여서 Custom URP render feature를 제작하여 적용

길찾기 알고리즘 연구

링크 :

https://github.com/Proffe..

플랫폼 : PC

사용 언어 : C#

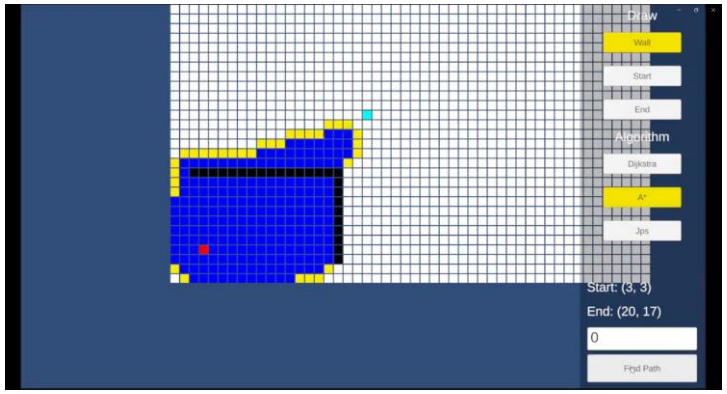
개발 인원 : 프로그래밍 1

개발 기간 : 2일

역할 : 메인 프로그래밍

#길찾기 알고리즘

#Astar #Jps




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

public abstract class BasePathFind : MonoBehaviour
{
    protected Coroutine routine;

    public void FindPathVisualize(Vector2Int start, Vector2Int end)
    {
        routine = StartCoroutine(FindPathRoutine(start, end));
    }

    public void StopPathFind()
    {
        if(routine != null)
            StopCoroutine(routine);
    }

    protected abstract IEnumerator FindPathRoutine(Vector2Int start, Vector2Int end);
}
```

● 프로젝트 설명

길찾기 알고리즘을
인 게임에 적용하기 위해
연구를 목적으로 만든 프로젝트

기존에 알고 있었던 Astar가 아닌 JPS
알고리즘에 대하여 호기심을 가지게 되었
고
코드를 구현하여 성능을 비교해 보았다.

● 활용 기술

- Astar, JPS

```cs
private void JumpDiagonal(Vector2Int start, Vector2Int dir, Vector2Int end, PriorityQueue<Point, float> pq, ref bool isFindEnd)
{
    var current = start;
    do
    {
        JumpStraight(current, new(dir.x, 0), end, pq, ref isFindEnd);
        if (isFindEnd) return;
        JumpStraight(current, new(0, dir.y), end, pq, ref isFindEnd);
        if (isFindEnd) return;

        current += dir;
        if (!map.ContainsCoord(current)) break;
        if (map[current].PointType == PointType.Wall) break;

        if (current == end)
        {
            map[end].Parent = map[start];
            map[end].IsJoined = true;
            isFindEnd = true;
            return;
        }

        if (current != start && current != end)
            map[current].Color = Color.gray;

        map[current].G = map[start].G + Vector2.Distance(start, current);
        map[current].Parent = map[start];

        var wall1 =
            map.ContainsCoord(new(current.x, current.y - dir.y)) &&
            map[new(current.x, current.y - dir.y)].PointType == PointType.Wall &&
            map.ContainsCoord(new(current.x + dir.x, current.y - dir.y)) &&
            map[new(current.x + dir.x, current.y - dir.y)].PointType != PointType.Wall;
        var wall2 =
            map.ContainsCoord(new(current.x - dir.x, current.y)) &&
            map[new(current.x - dir.x, current.y)].PointType == PointType.Wall &&
            map.ContainsCoord(new(current.x - dir.x, current.y + dir.y)) &&
            map[new(current.x - dir.x, current.y + dir.y)].PointType != PointType.Wall;
```