PROJECT4 : JELLY TAIL

PRESS THE BOTTON TO START

2023. 12. 11. MON | CHUNG-ANG UNIVERSITY

CONTENTS

001

WHAT

- description of SW system
- functionality

002

HOW

- overall design (UML)
- important implementation issue

003

DEMO

- program
- important functionality

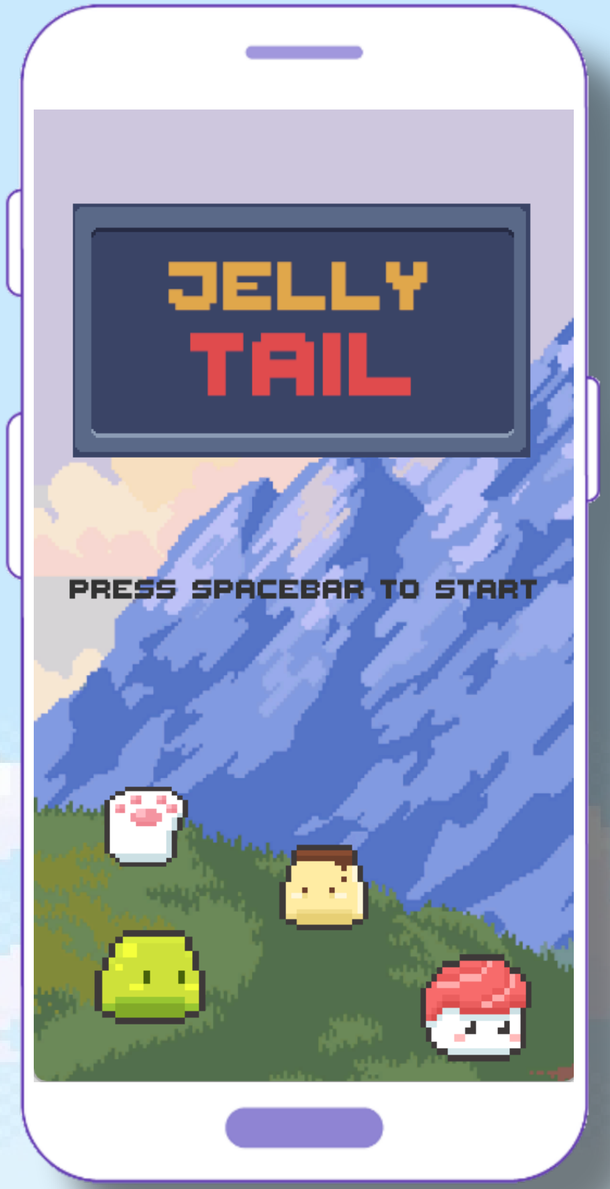
WHAT

- DESCRIPTION OF SW SYSTEM
- FUNCTIONALITY

001

- DESCRIPTION OF SW SYSTEM
- FUNCTIONALITY

001



- DESCRIPTION OF SW SYSTEM
- FUNCTIONALITY

001

- DESCRIPTION OF SW SYSTEM

- FUNCTIONALITY

001

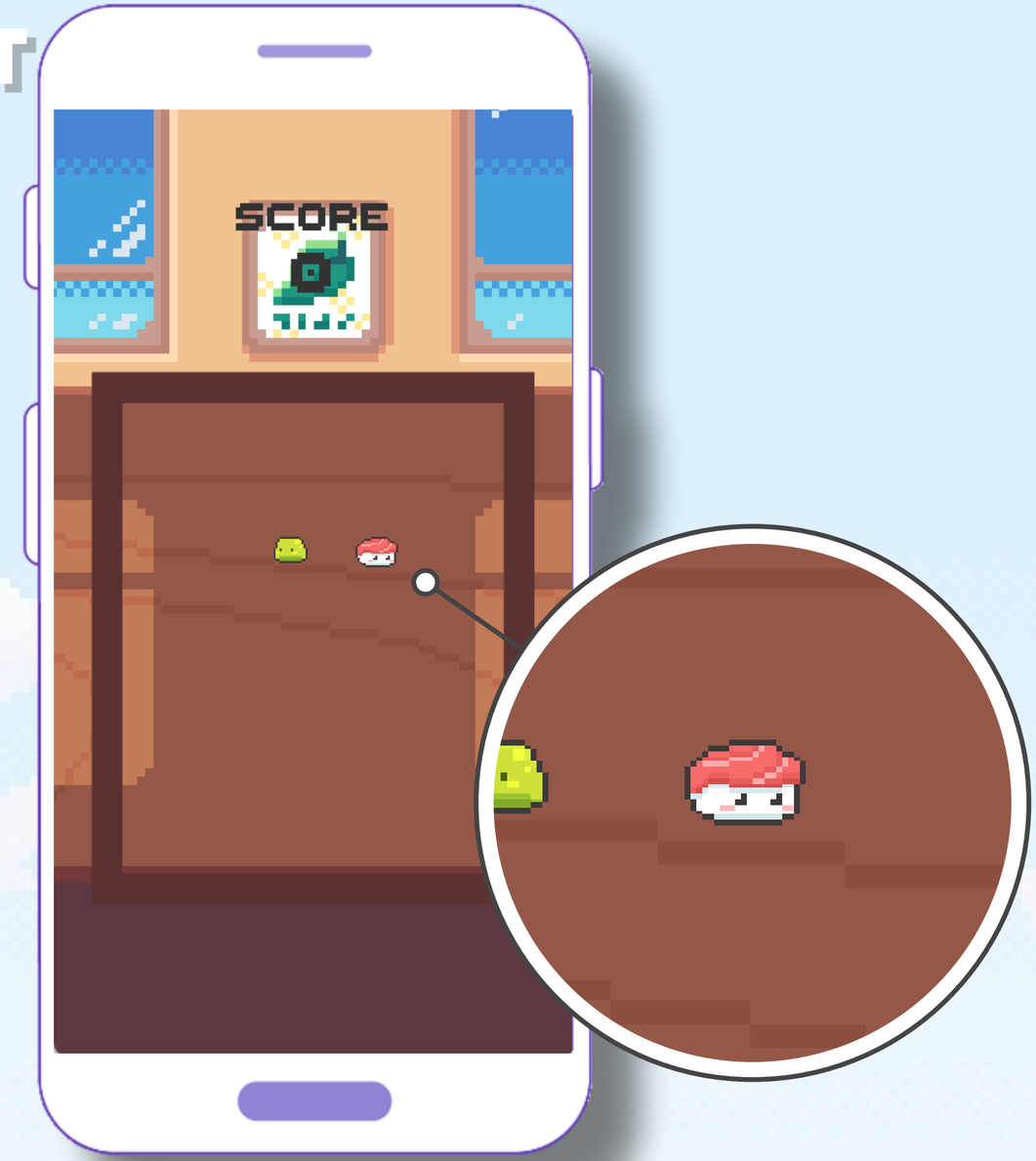
FUNCTIONALITY I : START


```
public class HeadMove : MonoBehaviour
{
    ...

    // Start is called before the first frame update
    void OnEnable()
    {

        currentRotationPoint = startRotationPos;
        objTransform.position = new Vector2(rotRadius, 0);
        PointInfo newinfo = new PointInfo();
        rotPointInfo.Add(newinfo);
        radius = head.GetComponent<CircleCollider2D>().radius;
        objTransform = head.GetComponent<Transform>();
        EventManager.Instance.AddTail += AddTail;
        EventManager.Instance.GameOver += GameOver;
        EventManager.Instance.GetHeadPos += GetHeadPos;

    }
```

```
public class TailMove : HeadMove
{
    public int pointIndex;

    private void OnEnable()
    {
        objTransform = GetComponent<Transform>();
    }
```

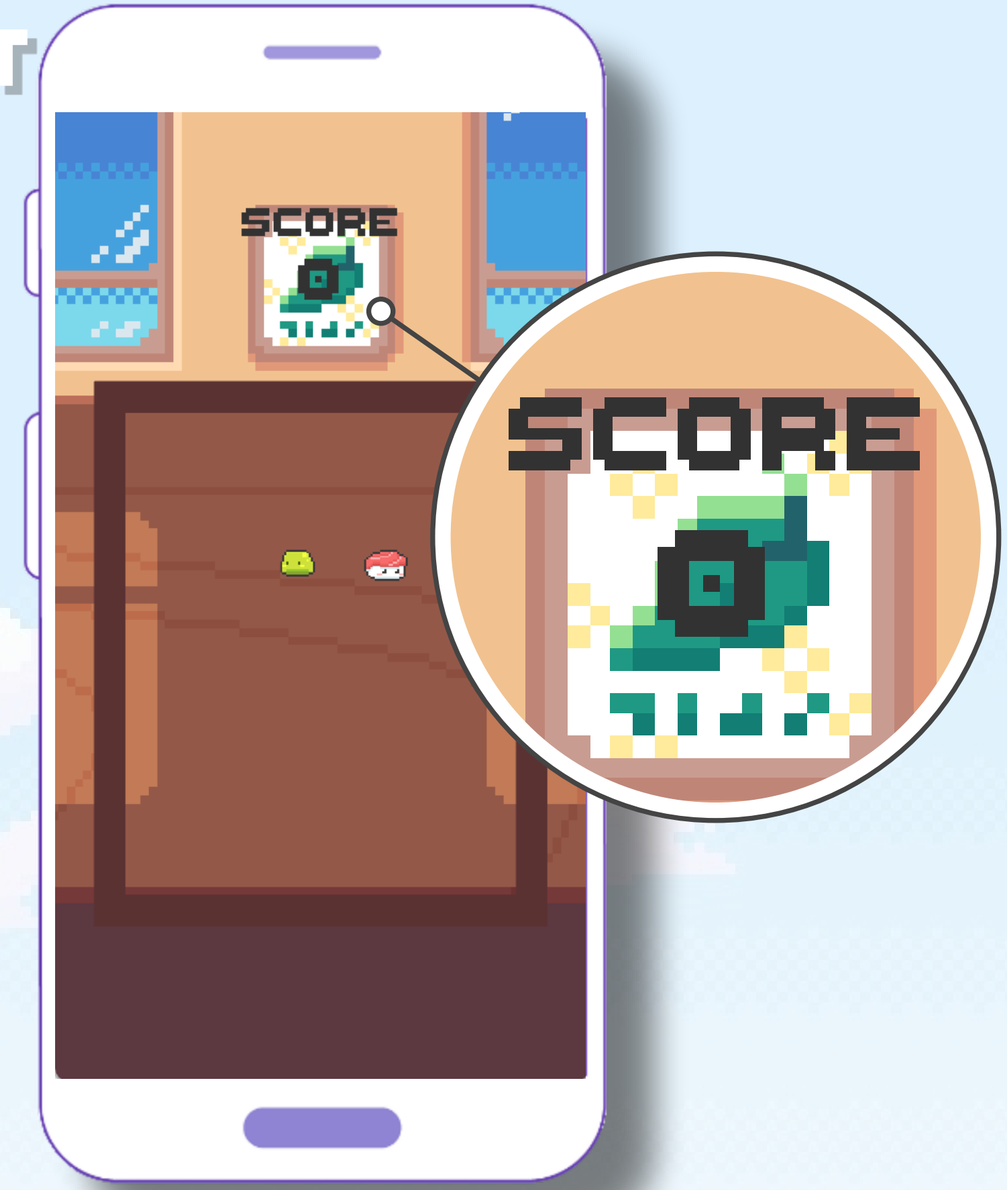
- DESCRIPTION OF SW SYSTEM
- FUNCTIONALITY

FUNCTIONALITY 1 : START




```
public class ItemManager : MonoBehaviour
{
    ...

    private void Start()
    {
        //이벤트 등록
        EventManager.Instance.CreateItem += CreateItem;
        EventManager.Instance.GameOver += GameOver;
        //초기 아이템 생성
        CreateItem();
    }
```

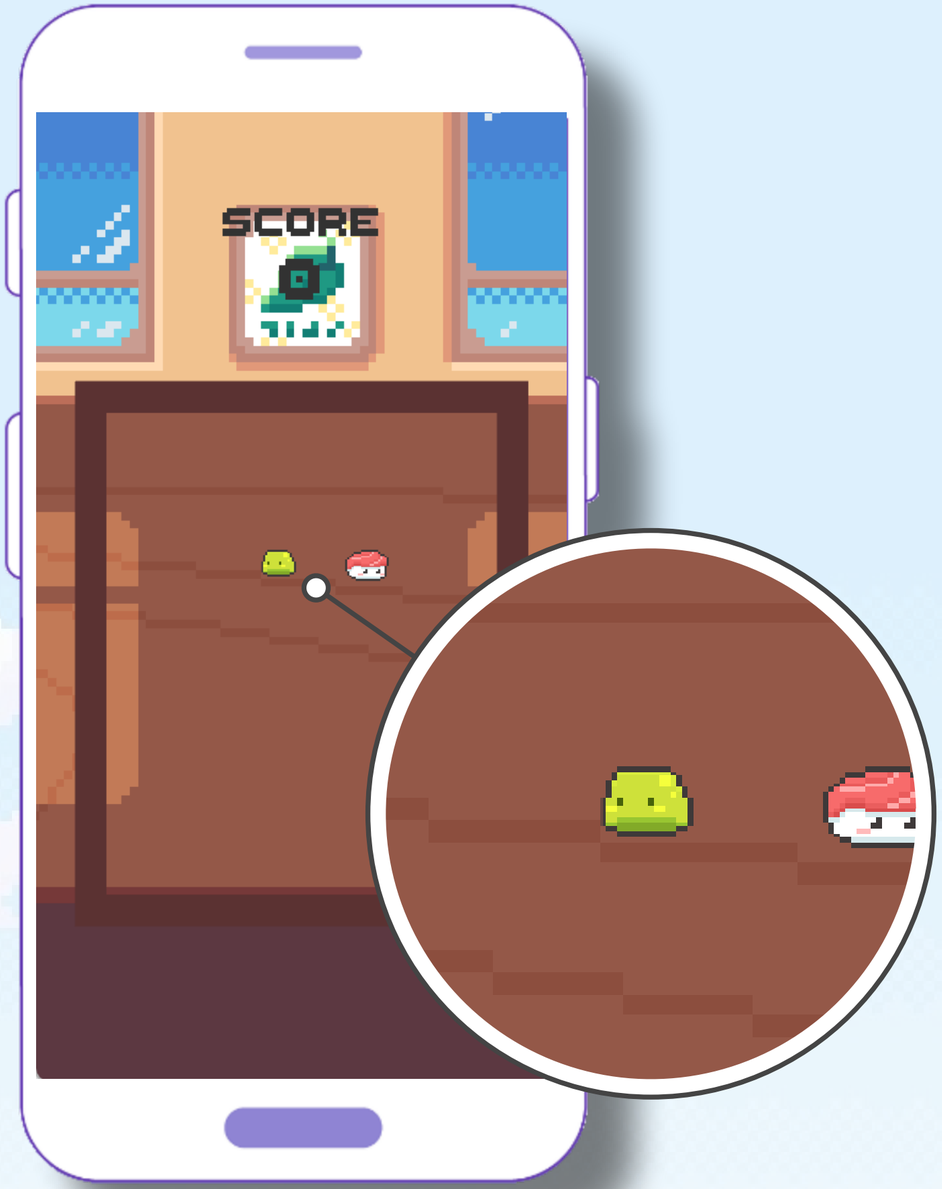
FUNCTIONALITY 1 : START




```
public class ScoreManager : MonoBehaviour
{
    [SerializeField] private int _bestScore;
    private static ScoreManager _instance;
    public static ScoreManager Instance => _instance;
    public void Awake()
    {
        if (_instance == null)
        {
            _instance = this;
            DontDestroyOnLoad(gameObject);
        }
        else Destroy(this);
    }

    private void Start()
    {
        EventManager.Instance.GetHighestScore += GetHighestScore;
        EventManager.Instance.UpdateScore += UpdateScore;
    }
```

```
public class GameUIController : MonoBehaviour
{
    ...

    int score = 0;
    int highestScore
    {
        get { return EventManager.Instance.CallOnGetHighestScore(); }
        set { EventManager.Instance.CallOnUpdateScore(value); }
    }

    private void Start()
    {
        SetScoreText();
        EventManager.Instance.GameOver += SetGameOver;
        EventManager.Instance.PlusScore += GetScore;
    }
```

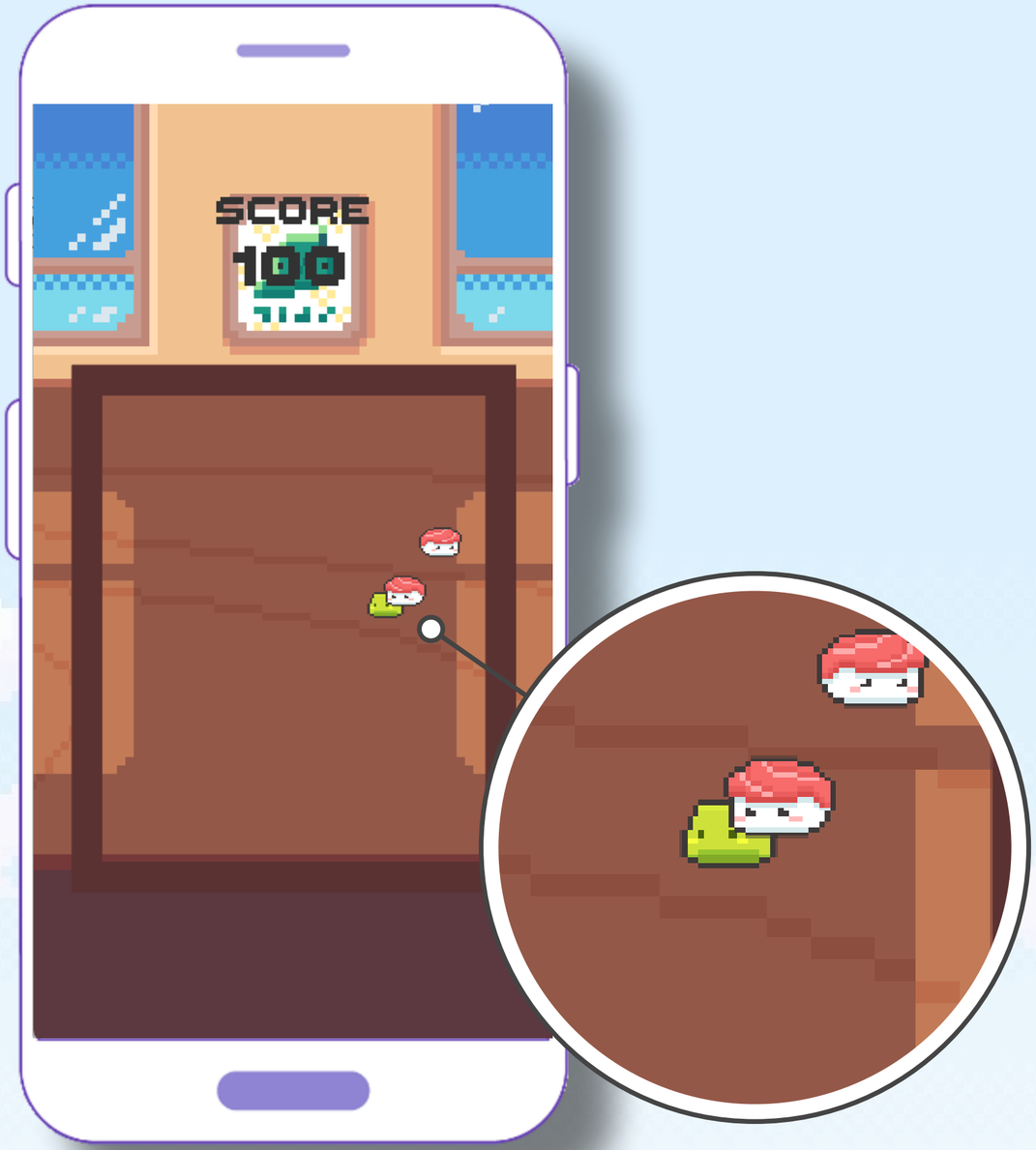
FUNCTIONALITY II: MOVE




```
public class HeadMove : MonoBehaviour
{
    ...

    private void Update()
    {
        flownTime += Time.deltaTime;
        Move();
        TailMove();

    }
```

```
private void Move()
{
    if (Input.GetKeyDown(KeyCode.Space))
    {
        //기존 마지막 회전점 정보에 끝날시간 입력
        rotPointInfo[rotPointInfo.Count() - 1].endTime = flownTime;
        //새로운 회전 중심 계산
        Vector2 newPoint = currentRotationPoint + 2 * ((Vector2)objTransform.position - currentRotationPoint);
        //방향 바꾸기
        isTurnClockwise = isTurnClockwise ? false : true;
        //회전 중심 변경
        currentRotationPoint = newPoint;
        //Vector2.right과 현재 중심에서 머리를 향하는 방향 사이의 각도(부호o)
        float nextAngle = Vector2.SignedAngle(Vector2.right, (Vector2)objTransform.position - currentRotationPoint);
        //새로운 회전점 정보 입력
        PointInfo newPointInfo = new PointInfo(isTurnClockwise, nextAngle, newPoint, flownTime);
        rotPointInfo.Add(newPointInfo);
    }
    //회전 방향결정
    Vector3 rotAxis = isTurnClockwise ? Vector3.forward : -Vector3.forward;
    //회전 중심을 기준으로 회전
    Vector2 newPos = Quaternion.AngleAxis(rotationSpeed * Time.deltaTime, rotAxis)
                   * ((Vector2)objTransform.position - currentRotationPoint);
    objTransform.position = currentRotationPoint + newPos;

    Debug.DrawLine(transform.position, currentRotationPoint, Color.blue);
}
```

FUNCTIONALITY II: MOVE




```
public class HeadMove : MonoBehaviour
{
    ...

    private void Update()
    {
        flownTime += Time.deltaTime;
        Move();
        TailMove();

    }
```

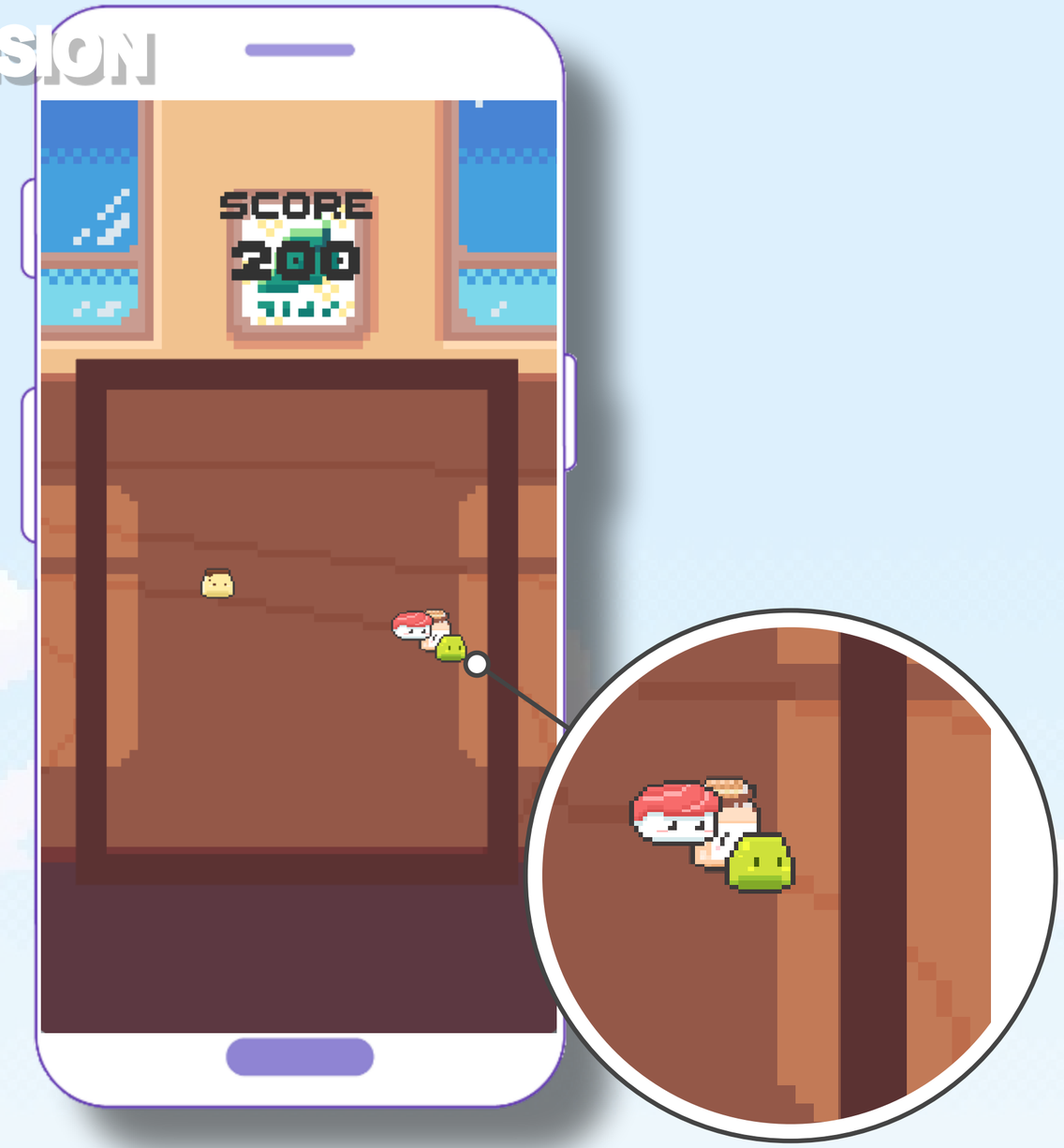
```
private void Move()
{
    if (Input.GetKeyDown(KeyCode.Space))
    {
        //기존 마지막 회전점 정보에 끝날시간 입력
        rotPointInfo[rotPointInfo.Count() - 1].endTime = flownTime;
        //새로운 회전 중심 계산
        Vector2 newPoint = currentRotationPoint + 2 * ((Vector2)objTransform.position - currentRotationPoint);
        //방향 바꾸기
        isTurnClockwise = isTurnClockwise ? false : true;
        //회전 중심 변경
        currentRotationPoint = newPoint;
        //Vector2.right과 현재 중심에서 머리를 향하는 방향 사이의 각도(부호o)
        float nextAngle = Vector2.SignedAngle(Vector2.right, (Vector2)objTransform.position - currentRotationPoint);
        //새로운 회전점 정보 입력
        PointInfo newPointInfo = new PointInfo(isTurnClockwise, nextAngle, newPoint, flownTime);
        rotPointInfo.Add(newPointInfo);
    }
    //회전 방향결정
    Vector3 rotAxis = isTurnClockwise ? Vector3.forward : -Vector3.forward;
    //회전 중심을 기준으로 회전
    Vector2 newPos = Quaternion.AngleAxis(rotationSpeed * Time.deltaTime, rotAxis)
                   * ((Vector2)objTransform.position - currentRotationPoint);
    objTransform.position = currentRotationPoint + newPos;

    Debug.DrawLine(transform.position, currentRotationPoint, Color.blue);
}
```

FUNCTIONALITY III: MOVE


```
private void TailMove()
{
    for(int i = 0; i < tails.Count; i++)
    {
        TailMove tailMoveScript = tails[i].GetComponent<TailMove>();
        PointInfo inputInfo = new PointInfo();
        int j = 0;
        for (; j < rotPointInfo.Count; j++)
        {
            if (tailMoveScript.flownTime >= rotPointInfo[j].startTime)
            {
                inputInfo = rotPointInfo[j];
            }
            else
                break;
        }
        if (tailMoveScript.pointIndex >= j)
            continue;
        else
        {
            tailMoveScript.SetInfo(inputInfo);
        }
    }
}
```

```
public class TailMove : HeadMove
{
    ...

    void Move()
    {
        //방향결정
        Vector3 rotAxis = isTurnClockwise ? Vector3.forward : -Vector3.forward;
        //회전 중심을 기준으로 회전
        Vector2 newPos = Quaternion.AngleAxis(rotationSpeed * Time.deltaTime, rotAxis)
                        * ((Vector2)objTransform.position - currentRotationPoint);
        objTransform.position = currentRotationPoint + newPos;
        Debug.DrawLine(transform.position, currentRotationPoint,Color.red);
    }
```

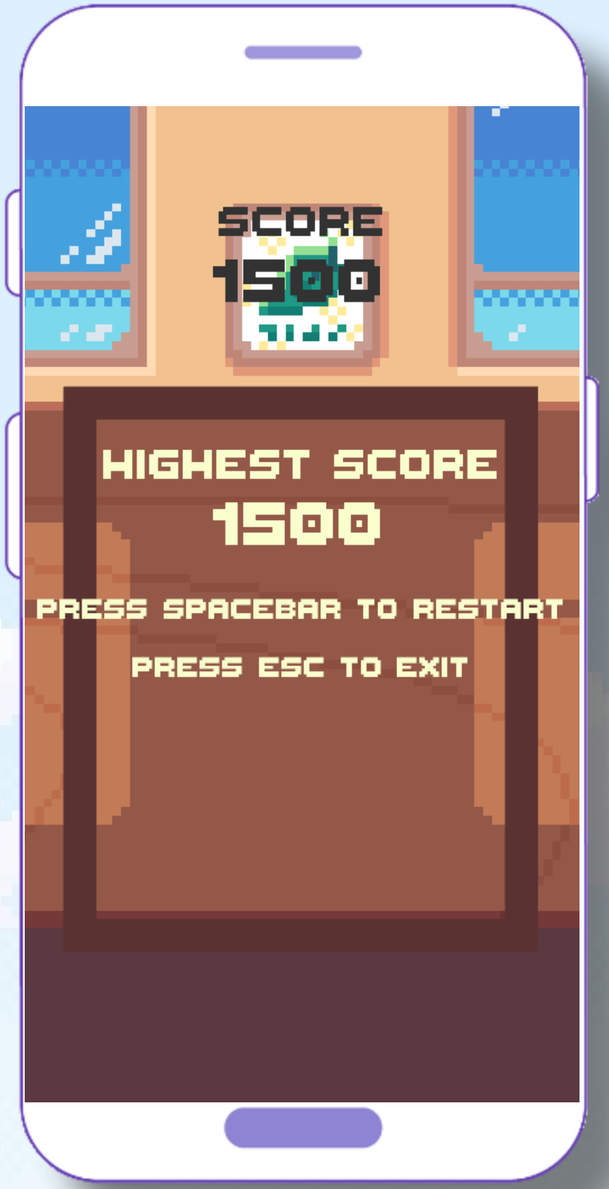
FUNCTIONALITY III: COLLISION




```
//충돌 입력 발생 시 실행
private void OnTriggerEnter2D(Collider2D collision)
{
    //게임종료 검사
    if (collision.gameObject.tag == "Tail")
    {
        if (moveScript.tails[0] != collision.gameObject)
        {
            EventManager.Instance.CallOnGameOver();
        }
    }
    if (collision.gameObject.tag == "Wall")
    {
        EventManager.Instance.CallOnGameOver();
    }

    // 아이템 획득 검사
    if (collision.gameObject.tag == "Item")
    {
        EventManager.Instance.CallOnPlusScore();
        GameObject tail = collision.GetComponent<ItemInfoHolder>().getTail;
        EventManager.Instance.CallOnAddTail(tail);
        Destroy(collision.gameObject);
        EventManager.Instance.CallOnCreateItem();
    }
}
```

- DESCRIPTION OF SW SYSTEM
- FUNCTIONALITY

FUNCTIONALITY IV: END




```
public void SetGameOver()
{
    if (!isGameFinished)
    {
        isGameFinished = true;
        flashingCoroutine = StartCoroutine(BlinkText());
    }


    if (highestScore < score) { highestScore = score; }
    Debug.Log(highestScore);

    highestLabel.text = "highest score";
    highestScoreText.text = highestScore.ToString();
}
```

HOW

- OVERALL DESIGN (UML)
- IMPORTANT IMPLEMENTATION
- ISSUE

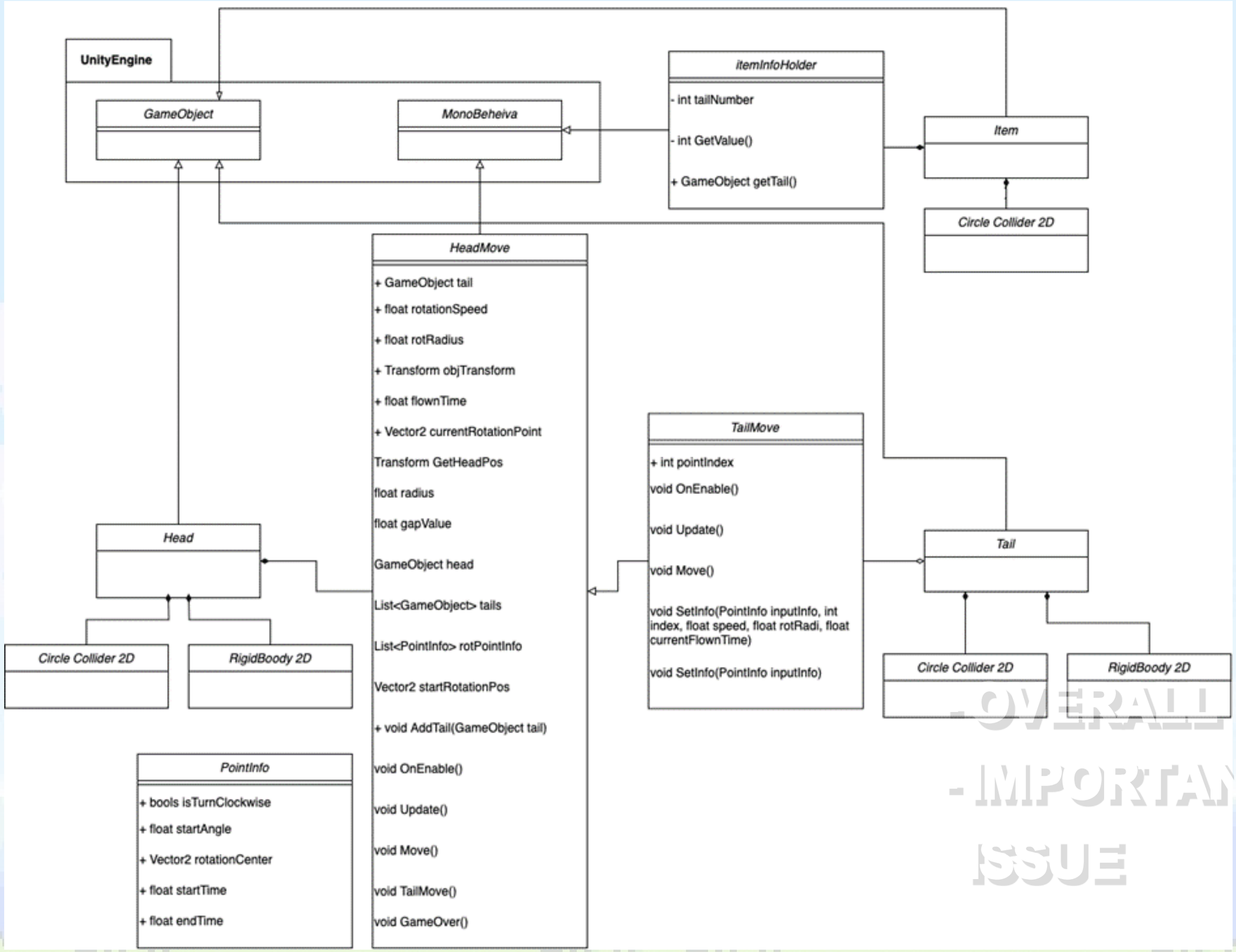
002

- OVERALL DESIGN (UML)
- IMPORTANT IMPLEMENTATION

ISSUE

002

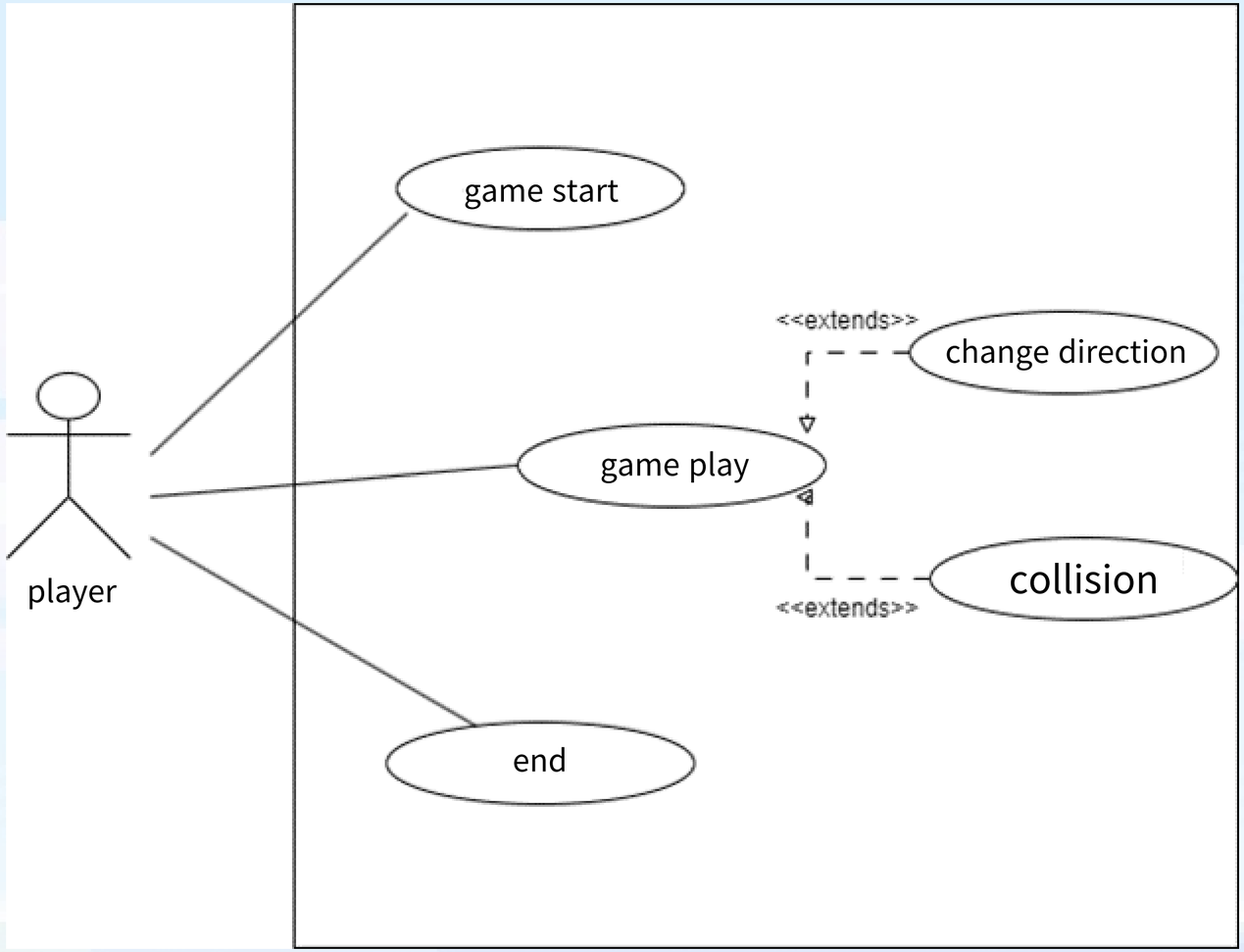
UML | : CLASS DIAGRAM



- OVERALL DESIGN (UML)
- IMPORTANT IMPLEMENTATION
- ISSUE

002

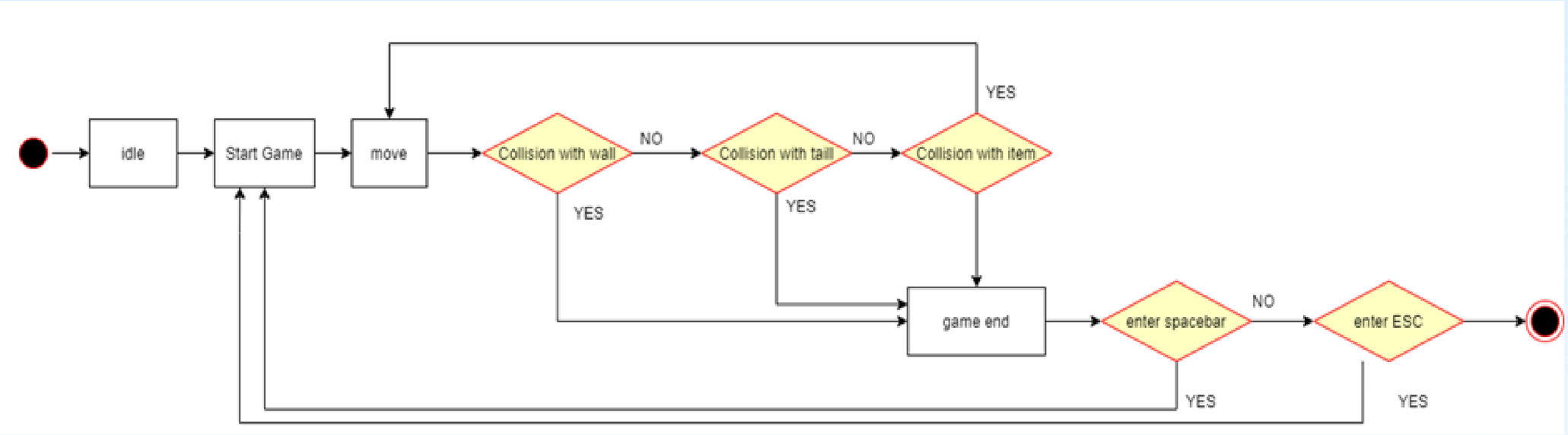
UML II:
USECASE DIAGRAM



- OVERALL DESIGN (UML)
- IMPORTANT IMPLEMENTATION ISSUE

002

UML III: ACTIVITY DIAGRAM



- OVERALL DESIGN (UML)
- IMPORTANT IMPLEMENTATION ISSUE

002

- OVERALL DESIGN (UML)
- IMPORTANT IMPLEMENTATION
- ISSUE

002

ISSUE I :
POINT INFO


```
public class PointInfo
{
    public PointInfo()
    {
        isTurnClockwise = true;
        startAngle = 0f;
        rotationCenter = Vector2.zero;
        startTime = 0f;
        endTime = int.MaxValue;
    }
    public PointInfo(bool rotateDir, float angle, Vector2 center, float sTime)
    {
        isTurnClockwise = rotateDir;
        startAngle = angle;
        rotationCenter = center;
        startTime = sTime;
        endTime = int.MaxValue;
}
    ~PointInfo() { }

    public bool isTurnClockwise;
    public float startAngle;
    public Vector2 rotationCenter;
    public float startTime;
    public float endTime;
}
```

- OVERALL DESIGN (UML)
- IMPORTANT IMPLEMENTATION

ISSUE 02

ISSUE II :
HEAD MOVE.CS


```
public class HeadMove : MonoBehaviour
{
    ...

    private void Update()
    {
        flownTime += Time.deltaTime;
        Move();
        TailMove();

    }
```

```
private void Move()
{
    if (Input.GetKeyDown(KeyCode.Space))
    {
        //기존 마지막 회전점 정보에 끝날시간 입력
        rotPointInfo[rotPointInfo.Count() - 1].endTime = flownTime;
        //새로운 회전 중심 계산
        Vector2 newPoint = currentRotationPoint + 2 * ((Vector2)objTransform.position - currentRotationPoint);
        //방향 바꾸기
        isTurnClockwise = isTurnClockwise ? false : true;
        //회전 중심 변경
        currentRotationPoint = newPoint;
        //Vector2.right과 현재 중심에서 머리를 향하는 방향 사이의 각도(부호o)
        float nextAngle = Vector2.SignedAngle(Vector2.right, (Vector2)objTransform.position - currentRotationPoint);
        //새로운 회전점 정보 입력
        PointInfo newPointInfo = new PointInfo(isTurnClockwise, nextAngle, newPoint, flownTime);
        rotPointInfo.Add(newPointInfo);
    }
    //회전 방향결정
    Vector3 rotAxis = isTurnClockwise ? Vector3.forward : -Vector3.forward;
    //회전 중심을 기준으로 회전
    Vector2 newPos = Quaternion.AngleAxis(rotationSpeed * Time.deltaTime, rotAxis)
                   * ((Vector2)objTransform.position - currentRotationPoint);
    objTransform.position = currentRotationPoint + newPos;

    Debug.DrawLine(transform.position, currentRotationPoint, Color.blue);
}
```

- OVERALL DESIGN (UML)
- IMPORTANT IMPLEMENTATION

ISSUE 02

ISSUE II : HEAD MOVE.CS


```
void TailMove()
{
    for(int i = 0; i < tails.Count; i++)
    {
        TailMove tailMoveScript = tails[i].GetComponent<TailMove>();
        PointInfo inputInfo = new PointInfo();
        int j = 0;
        for (; j < rotPointInfo.Count; j++)
        {
            if (tailMoveScript.flownTime >= rotPointInfo[j].startTime)
            {
                inputInfo = rotPointInfo[j];
            }
            else
                break;
        }
        if (tailMoveScript.pointIndex >= j)
            continue;
        else
        {
            tailMoveScript.SetInfo(inputInfo);
        }
    }
}
```

```
public void AddTail(GameObject tail)
{
    //꼬리 오브젝트 생성하면서 현재 꼬리 오브젝트 개수를 기반으로 계산한
    //생성될 시간을 바탕으로 PointInfo 입력.

    //필요 최소 각도(deg)
    float theta = Mathf.Asin(radius / rotRadius) * 2 * Mathf.Rad2Deg;

    //얼마나 전 위치에 넣어야 하는지 시간 계산
    float timeBefore = (tails.Count() + 1) * (theta / rotationSpeed) * gapValue;
    //가능성 적긴 하지만
    if (timeBefore >= flownTime)
    {
        timeBefore = flownTime;
    }
    //Debug.Log("최소 각도 = "+ theta +", " + timeBefore + "초 전에 넣음.");
    PointInfo returnInfo = new PointInfo();
    int i = 0;
    for (; i < rotPointInfo.Count; i++)
    {
        //flownTime - timeBefore = 머리가 이 순간에 위치한 좌표가 꼬리가 위치할 좌표
        if (rotPointInfo[i].startTime <= flownTime - timeBefore) { returnInfo = rotPointInfo[i]; }
        else { break; }
    }
    TailMove newTailScript = tail.GetComponent<TailMove>();
    newTailScript.SetInfo(returnInfo, i, rotationSpeed, rotRadius, flownTime - timeBefore);
    tails.Add(tail.gameObject);
}
```

- OVERALL DESIGN (UML)
- IMPORTANT IMPLEMENTATION

ISSUE 02

ISSUE III:
TAIL MOVE.CS


```
public class TailMove : HeadMove
```

```
void Update()
{
    flownTime += Time.deltaTime;
    Move();
}
```

```
public void SetInfo(PointInfo inputInfo, int index, float speed, float rotRadi, float currentFlownTime)
{
    pointIndex = index;
    if (speed != int.MaxValue)
    {
        rotationSpeed = speed;
    }
    if(rotRadi != int.MaxValue)
    {
        rotRadius = rotRadi;
    }
    isTurnClockwise = inputInfo.isTurnClockwise;
    currentRotationPoint = inputInfo.rotationCenter;
    //위치보정용 시간차이
    float timeDiff = currentFlownTime - inputInfo.startTime;
    flownTime = currentFlownTime;
    //회전 방향로 보정값 부호 결정
    timeDiff *= isTurnClockwise ? 1 : -1;
    //보정각도 계산(처음각도 + 시간차이로 보정한 각도)
    float deltaAngle = rotationSpeed * timeDiff + inputInfo.startAngle;
    //위치 계산
    Vector2 newPos = (Quaternion.AngleAxis(deltaAngle, Vector3.forward) * Vector2.right) * rotRadius;
    //위치 입력
    transform.position = currentRotationPoint + newPos;
}
```

- OVERALL DESIGN (UML)
- IMPORTANT IMPLEMENTATION

ISSUE 02

ISSUE III:
TAIL MOVE.CS


```
public void SetInfo(PointInfo inputInfo)
{
    //회전 중심 순서 1 증가
    pointIndex++;
    //회전 방향 재설정
    isTurnClockwise = inputInfo.isTurnClockwise;
    //회전 중심 재설정
    currentRotationPoint = inputInfo.rotationCenter;
    //위치보정용 시간차이 계산
    float timeDiff = flownTime - inputInfo.startTime;
    //회전 방향로 보정값 부호 결정
    timeDiff *= isTurnClockwise ? 1 : -1;
    //보정각도 계산(처음각도 + 시간차이로 보정한 각도)
    float deltaAngle = rotationSpeed * timeDiff + inputInfo.startAngle;
    //위치 계산
    Vector2 newPos = (Quaternion.AngleAxis(deltaAngle, Vector3.forward) * Vector2.right) * rotRadius;
    //위치 입력
    transform.position = currentRotationPoint + newPos;
}
```

ISSUE Ⅳ:
COLLISION CONTROLLER.CS


```
public class HeadCollisionController : MonoBehaviour
{
    [SerializeField] HeadMove moveScript;

    //충돌 입력 발생 시 실행
    private void OnTriggerEnter2D(Collider2D collision)
    {
        //게임종료 검사
        if (collision.gameObject.tag == "Tail")
        {
            if (moveScript.tails[0] != collision.gameObject)
            {
                EventManager.Instance.CallOnGameOver();
            }
        }
        if (collision.gameObject.tag == "Wall")
        {
            EventManager.Instance.CallOnGameOver();
        }

        // 아이템 획득 검사
        if (collision.gameObject.tag == "Item")
        {
            EventManager.Instance.CallOnPlusScore();
            GameObject tail = collision.GetComponent<ItemInfoHolder>().getTail;
            EventManager.Instance.CallOnAddTail(tail);
            Destroy(collision.gameObject);
            EventManager.Instance.CallOnCreateItem();
        }
    }
}
```

ISSUE V:
ITEM INFO HOLDER & ITEM MANAGER

2023. 12. 11. MON | CHUNG-ANG UNIVERSITY

```
public class ItemInfoHolder : MonoBehaviour
{
    [SerializeField] GameObject[] tails;
    [SerializeField] int tailNumber;

    public GameObject getTail { get { return Instantiate(tails[tailNumber]);} }
}
```

```
public class ItemManager : MonoBehaviour
{
    [SerializeField] List<GameObject> _items = new List<GameObject>();
    // 아이템 생성 범위 range
    [SerializeField] private float _up;
    [SerializeField] private float _down;
    [SerializeField] private float _left;
    [SerializeField] private float _right;
    [SerializeField] private float _distance;

    ...

    //아이템 프리팹 생성
    private GameObject GetItem()
    {
        var index = Random.Range(0, _items.Count);
        GameObject i = Instantiate(_items[index]);
        return i;
    }

    //아이템 생성, 랜덤 위치에 배치
    public void CreateItem()
    {
        Vector3 position = new Vector3(Random.Range(_left, _right), Random.Range(_down, _up), 0);

        //현재 플레이어의 위치와의 거리가 기준 이하로 가까울 경우 위치 조정
        if (Vector3.Magnitude(position - EventManager.Instance.CallOnGetHeadPos().position) < _distance + 0.01f)
        {
            position += new Vector3(_distance, 0, 0);
            if (position.x > _right)
            {
                position = new Vector3(_left + 0.5f, position.y, 0);
            }
        }

        GameObject item = GetItem();
        _item = item;
        item.transform.position = position;

    }

    private void GameOver()
    {
        //게임 종료 시 현재 존재하는 아이템 오브젝트 파괴
        if (_item != null) { Destroy(_item); }
    }
}
```

- OVERALL DESIGN (UML)
- IMPORTANT IMPLEMENTATION

ISSUE 002

DEMO

- PLAY PROGRAM

003

PLAY PROGRAM

003

QNA

2023. 12. 11. MON I OOP I TEAM 8

20184256 박성민
20226041 김규리
20170530 장기봉
20220127 유선기