📌 화면 이동 방식에 따른 카메라 무브 기획서

1. 개요

문서명: 화면 이동 방식에 따른 카메라 무브
목적: 이 기획서를 통해 개발팀에서 게임 내 씬 이동 방식을 확인하고 제작 방향을 결정한다.

2. 주요 내용

게임 내에서 유저가 홀과 주방, 그리고 층간을 이동하는 방식에 대해 두 가지 방법을 개발팀에 제안한다.

3. 화면 이동 방식 제안

1) 버튼을 활용한 화면 이동 방식

버튼을 눌러 특정 구역으로 이동하는 방식
- 1-1: 화면 내 버튼을 이용하여 홀과 주방을 넘나들게 한다.
- 1-2: 기존에 사용된 층간 이동 버튼을 활용하여, 이후 오픈될 층간 이동도 버튼으로 해결한다.

장점:
- 직관적인 인터페이스로 유저가 쉽게 조작 가능
- 특정 구역으로 즉시 이동이 가능하여 빠른 전환이 이루어짐
- 기존 버튼 시스템을 활용하여 개발 부담 최소화

단점:
- 연속적인 이동이 아닌 단계별 이동 방식이므로 화면 전환이 다소 단절될 수 있음
- 드래그를 통한 자연스러운 움직임이 불가능

---

2) 버튼 & 화면 슬라이드 방식

터치 슬라이드(드래그) + 버튼을 혼합한 방식
- 2-1: 버튼과 화면 터치를 슬라이드 방식으로 사용하여 좌우로 이동한다.
- 2-2: 기존 버튼 외에도 직접 화면을 터치 & 드래그하여 주방과 홀, 그리고 위층과 아래층을 이동할 수 있다.
- 2-3:
- 화면 내 터치 요소만 존재 (예: 돈, 쓰레기, 주문받기).
- 메인 UI 버튼 영역을 제외한 빈 화면에서 드래그하면 이동.
- 카메라가 일정 영역을 넘어가면 자동으로 다음 구역(주방↔홀, 1층↔2층)으로 X축 & Y축 이동.

장점:
- 직관적인 조작 방식으로 모바일 환경에 최적화됨
- 연속적인 이동이 가능하여 더 부드러운 게임 경험 제공
- 기존 버튼 방식보다 자연스럽게 씬 전환 가능

단점:
- 기존 UI와 충돌 가능성이 있어 드래그 가능 영역을 명확하게 설정해야 함
- 버튼 단독 이동 방식보다 조작 난이도가 약간 증가할 수 있음

---

4. 결론 및 선택 방향

버튼 방식 vs 버튼+슬라이드 방식 비교 분석

방식 직관성 조작 난이도 개발 난이도화면 전환 속도UX 자연스러움
버튼 이동 방식 높음 낮음 낮음 빠름다소 단절적
버튼 + 슬라이드높음 중간 중간 중간자연스러움

개발팀에서는 유저 경험(UX) 및 기술 구현 가능성을 고려하여 두 가지 방식 중 적절한 이동 방식을 결정해야 한다.

5. 최종 결정 시 고려할 요소

슬라이드 방식 적용 시:
- UI 요소(돈, 쓰레기 클릭 등)와 충돌이 발생하지 않도록 예외 처리 필요
- X축 & Y축 이동 범위 설정을 명확하게 해야 함

버튼 방식 적용 시:
- 직관적인 사용성을 유지하면서도 추후 확장성을 고려해야 함

최적의 방향을 결정하여 개발 진행

📌유니티에서 슬라이드 방식의 카메라 무브 구현 및 구조 설계

1. 개요

현재 버튼을 활용하여 씬을 전환하는 기존 방식에서 벗어나, 슬라이드 방식을 사용하여 카메라가 특정 포커스를 넘어가면 다른 영역으로 이동하는 방식으로 변경하고자 한다. 이를 위해 유니티에서 카메라 이동 시스템을 어떻게 설계하고 구현할지에 대한 가이드라인을 제공한다.

---

2. 카메라 이동 방식 설계

A. 기본적인 카메라 이동 구조

1. 플레이어가 화면을 드래그하면 X축 또는 Y축 방향으로 카메라 이동
2. 카메라의 중심이 특정 임계점을 초과하면 새로운 영역(홀↔주방, 1층↔2층)으로 자동 이동
3. 슬라이드 중간에 포커스가 특정 영역에 도달하면 자동으로 정렬하여 이동을 멈춤
4. 메인 UI 버튼(코인, 쓰레기, 주문 등) 영역은 터치 이벤트에서 제외
5. 팝업이 활성화되었을 경우, 화면 슬라이드가 비활성화됨

---

3. 코드 설계 (C# 유니티 구현 예시)

A. 카메라 이동을 위한 주요 변수 및 설정

```
using UnityEngine;

public class CameraController : MonoBehaviour
{
    public Transform[] targetPositions; // 이동 가능한 영역들의 위치 리스트 (주방, 홀, 1층, 2층 등)
    public float snapSpeed = 5f; // 카메라가 부드럽게 이동하는 속도
    public float threshold = 1.5f; // 새로운 영역으로 이동하는 임계값

    private Vector3 dragStartPos;
    private Vector3 cameraStartPos;
    private bool isDragging = false;
    private int currentTargetIndex = 0;
    private bool isPopupActive = false; // 팝업 활성화 여부
}
```

B. 드래그 입력을 받아 카메라 이동 처리

```
void Update()
{
    if (isPopupActive) return; // 팝업이 활성화되었으면 입력 차단

    if (Input.GetMouseButtonDown(0))
    {
        if (IsPointerOverUIElement()) return; // UI 버튼 클릭 시 카메라 이동 방지

        isDragging = true;
        dragStartPos = Camera.main.ScreenToWorldPoint(Input.mousePosition);
        cameraStartPos = transform.position;
    }

    if (Input.GetMouseButton(0) && isDragging)
    {
        Vector3 difference = dragStartPos - Camera.main.ScreenToWorldPoint(Input.mousePosition);
        transform.position = cameraStartPos + new Vector3(difference.x, 0, 0);
    }

    if (Input.GetMouseButtonUp(0))
    {
        isDragging = false;
        CheckForSnap();
    }
}
```

C. 카메라 포커스 변경 및 스냅 이동 (자동 정렬)

```
void CheckForSnap()
{
   for (int i = 0; i < targetPositions.Length; i++)
   {
      if (Mathf.Abs(transform.position.x - targetPositions[i].position.x) < threshold)
      {
         currentTargetIndex = i;
         StartCoroutine(SmoothMove(targetPositions[i].position));
         break;
      }
   }
}

IEnumerator SmoothMove(Vector3 targetPosition)
{
   while (Vector3.Distance(transform.position, targetPosition) > 0.1f)
   {
      transform.position = Vector3.Lerp(transform.position, targetPosition, snapSpeed * Time.deltaTime);
      yield return null;
   }
   transform.position = targetPosition;
}
```

D. UI 요소에서 드래그 이벤트 차단

```
using UnityEngine.EventSystems;

bool IsPointerOverUIElement()
{
   return EventSystem.current.IsPointerOverGameObject();
}
```

- 설명:

IsPointerOverUIElement()를 사용하여 현재 마우스 또는 터치 입력이 UI 버튼 영역에서 발생하면 카메라 이동을 차단한다.

E. 팝업 활성화 시 카메라 이동 차단

```
public void ShowPopup()
{
   isPopupActive = true;
}

public void HidePopup()
{
   isPopupActive = false;
}
```

- 설명:

팝업이 활성화된 동안 isPopupActive를 true로 설정하여 Update()에서 카메라 이동을 차단한다.

---

4. 결론 및 최적화 방향

✅ 드래그를 통한 자연스러운 이동을 가능하게 하면서도 자동 포커스 조정 기능을 추가하여 불편함을 줄인다.
✅ UI 버튼(코인, 쓰레기, 주문 등) 클릭 시에는 카메라 이동을 방지하여 조작 오류를 막는다.
✅ 팝업이 활성화되면 슬라이드 기능을 비활성화하여 사용자 경험을 개선한다.
✅ X축 & Y축 이동을 분리하여 특정 구역을 초과하면 자동 이동하는 방식을 유지한다.