게임엔진프로그래밍

ANIMAL COLLECTOR

2024.06.13.

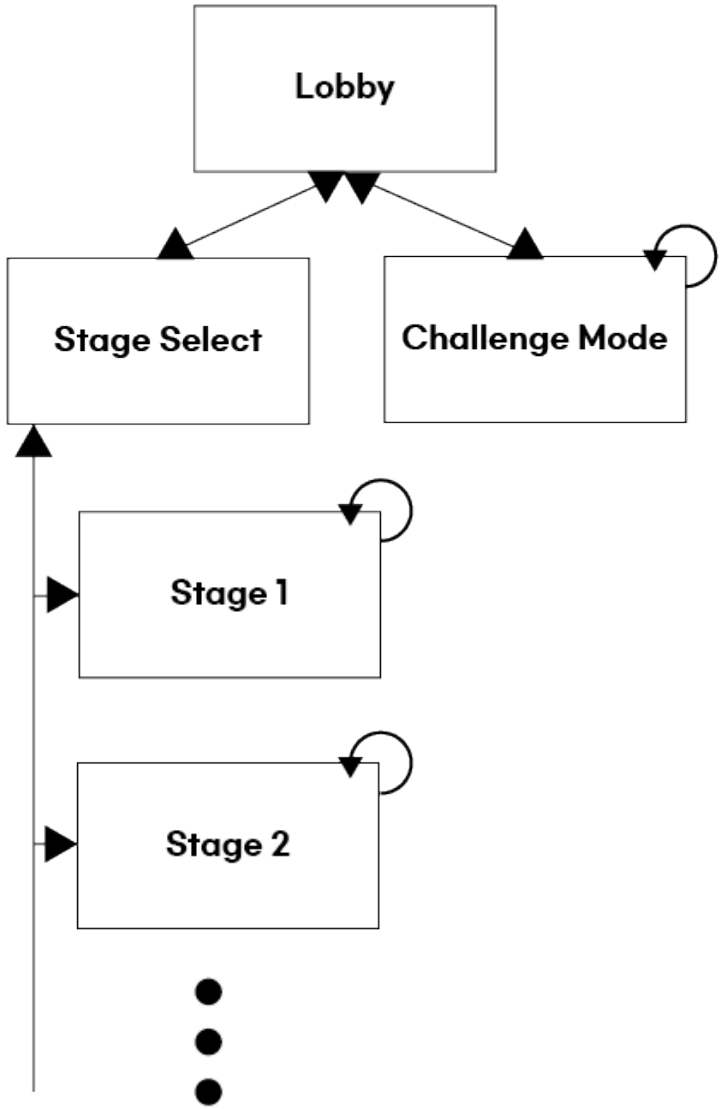
디지털미디어학과(김현정)

프로젝트 요약

- **프로젝트 요약**
    - 스테이지에서 목표점수보다 높은 점수를 얻게 되면, 각 스테이지마다 정해진 동물이 로비에 등장하는 수집형 퍼즐 게임
    - 드래그 영역 내의 오브젝트들 숫자가 도합 10이 되도록 선택하여 파괴하면 점수를 얻는 것이 기본 룰이다.
    - 스테이지 모드와 챌린지 모드가 존재하며, 스테이지 모드는 오브젝트의 숫자가 고정, 챌린지 모드는 랜덤이다.
- **프로그래밍과정에서 주안점**
    - 드래그를 하고 있을 때, 선택된 오브젝트들이 눈에 띄도록 해야 함.
    - 각 스테이지에서 플레이어가 목표를 성공했는 지에 대한 여부를 가져와 로비에서 사용해야 함.
    - 각 스테이지 별로 최고 점수를 기록하여 이전 자신의 점수를 볼 수 있어야 함.
    - 선택된 오브젝트들의 합이 10이 되면 제거. 10의 배수여도 안됨.

프로젝트 수행 결과물

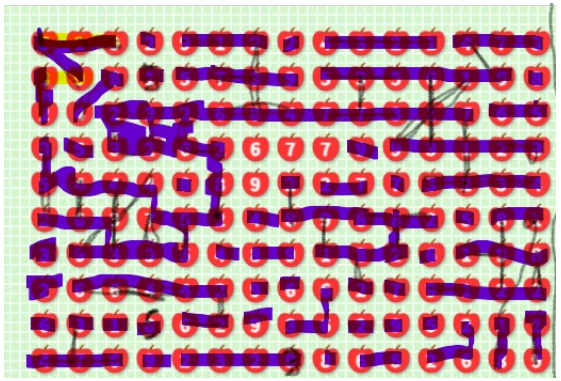
- **게임 설계**
  - 각 씬의 이동 가능 범위 가시화.
- **씬 별 이동 정리**
  - Lobby
  - 스테이지 선택 화면, 챌린지 모드로 이동 가능.
  - 게임 종료 버튼도 존재.
  - Stage Select
  - 스테이지를 고를 수 있으며, 로비로도 이동 가능.
  - Challenge Mode
  - 새로고침 가능(게임 종료 시), 로비로 이동 가능.
  - Stage
  - 새로고침 가능(게임 종료 시), 스테이지 선택 화면으로 이동 가능

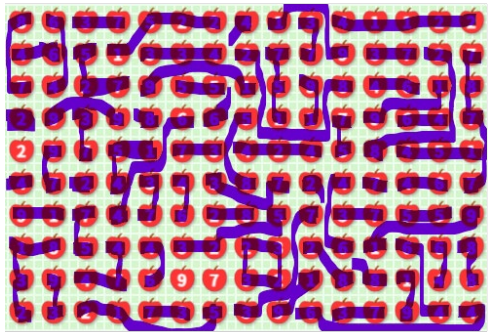


스테이지는 로직이 같으므로 추가 가능

프로젝트 수행 결과물

- **스테이지 설계**
  - 스테이지는 맵이 고정되기 때문에 혹시나 다 외워서 모든 오브젝트를 다 부수는 경우를 상정.
  - FruitBox라는 유사 게임으로부터 랜덤 생성된 오브젝트를 가져와 수정.

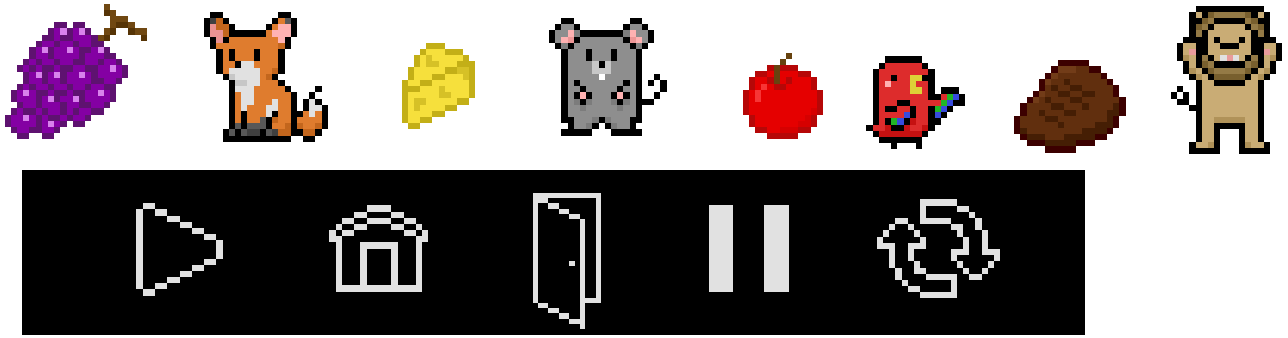


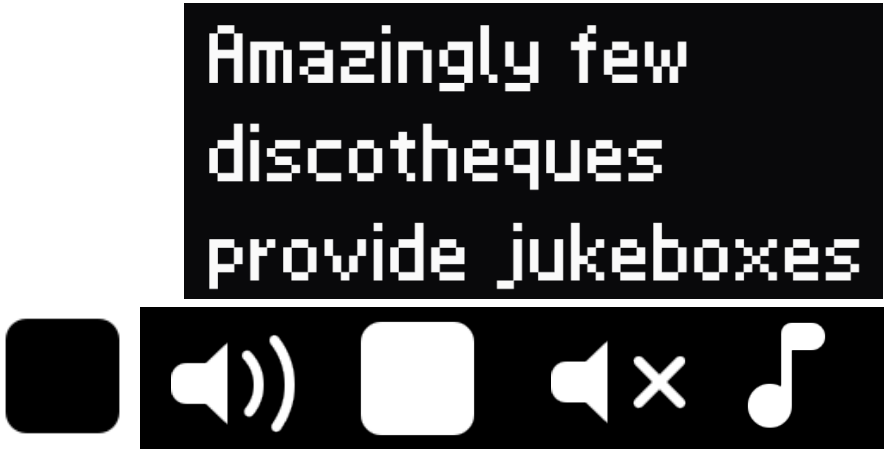


```
[SerializeField]
GameObject blockPrefab;
[SerializeField]
int stage = 0;

private List<GameObject> blockList = new List<GameObject>( );
int[] StageOneNum = {1,1,9,2,9,1,2,2,4,3,2,1,4,7,3,
                     8,9,6,5,1,7,3,9,5,3,5,6,1,6,8,
                     2,4,2,3,2,6,8,4,7,7,3,1,7,6,4,
                     3,5,3,2,6,2,6,4,1,5,9,3,1,2,8,
                     5,4,4,4,8,8,9,2,2,8,7,3,2,3,4,
                     2,6,3,7,5,2,4,8,9,8,8,2,5,8,1,
                     3,6,3,3,5,4,8,1,1,8,2,5,1,9,8,
                     2,6,6,4,4,2,1,6,6,2,5,9,1,2,2,
                     6,3,1,5,6,2,3,3,1,2,7,5,2,1,1,
                     4,7,6,9,2,3,2,3,8,6,4,2,6,8,9};
int[] StageTwoNum = {3,6,3,7,5,2,2,4,1,3,4,1,1,2,2,
                     4,6,5,1,3,3,4,2,7,7,9,3,7,2,4,
                     7,4,2,3,9,5,5,1,2,9,8,4,6,1,3,
                     2,1,3,9,8,6,6,5,1,1,7,9,6,4,2,
                     2,3,7,5,1,7,3,4,2,4,3,8,4,5,1,
                     2,7,1,4,3,4,3,5,7,2,4,7,4,6,1,
                     9,1,5,4,7,6,2,8,5,7,3,7,5,5,9,
                     1,4,5,4,4,3,2,2,3,2,6,1,6,6,5,
                     3,9,4,2,6,5,3,3,2,1,8,8,1,4,5,
                     2,1,2,3,7,3,5,3,9,6,3,7,9,4,4};
```

프로젝트 수행 결과물

- **제작 리소스 사용**
  - Piskel을 통해 도트 이미지 제작



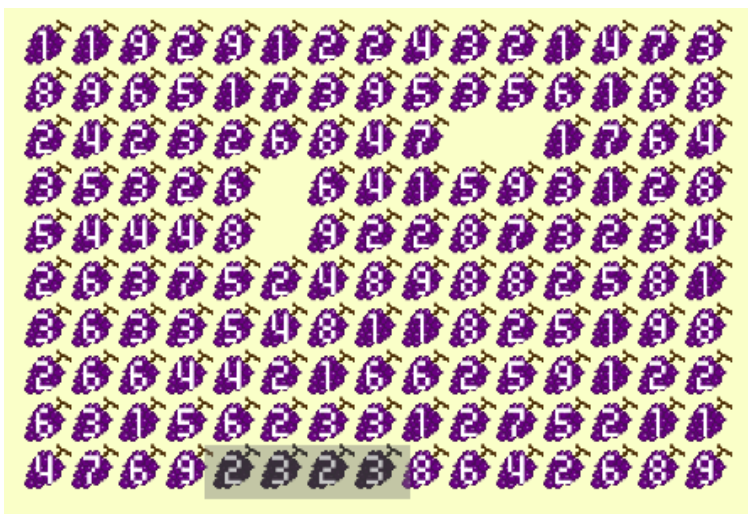


- **외부 리소스 사용**
  - Font : Galmuri7(https://galmuri.quiple.dev/)
  - Icons8(https://icons8.com/)
  - BGM : https://pixabay.com/music/video-games-bit-beats-3-168873/
  - 효과음 : SFX provided by 셀바이뮤직(https://sellbuymusic.com/md/savicbb-vczcfck)

프로젝트 수행 결과물

- **게임 로직 구현**
  - 마우스 드래그 영역 가시화.
  - 마우스 버튼이 눌리면 드래그 영역을 나타낼 사각형이 생성됨
  - 마우스 버튼이 눌리고 있는 동안에는 눌렸을 때의 마우스 포인터 위치와 현재 마우스 포인터 위치를 계산하여 드래그 영역 업데이트
  - 마우스 버튼이 떼어지면 영역을 나타내던 사각형 제거

```
Unity 메시지 | 참조 0개
void Update()
{
    if(mouseActive == true)
    {
        if (Input.GetMouseButtonDown(0))
        {
            startPos = Camera.main.ScreenToWorldPoint(new Vector3(Input.mousePosition.x, Input.mousePosition.y, Camera.main.transform.position.z * -1));
            square = Instantiate(dragSquare, new Vector3(0, 0, 0), Quaternion.identity);
        }

        if (Input.GetMouseButton(0))
        {
            nowPos = Camera.main.ScreenToWorldPoint(new Vector3(Input.mousePosition.x, Input.mousePosition.y, Camera.main.transform.position.z * -1));
            deltaX = Mathf.Abs(nowPos.x - startPos.x);
            deltaY = Mathf.Abs(nowPos.y - startPos.y);
            deltaPos = startPos + (nowPos - startPos) / 2;
            square.transform.position = deltaPos;
            square.transform.localScale = new Vector3(deltaX, deltaY, 0);
        }

        if (Input.GetMouseButtonUp(0))
        {
            Destroy(square);
        }
    }
    else
    {
        Destroy(square);
    }
}
```



↑ : 드래그 영역

프로젝트 수행 결과물

- **게임 로직 구현**
  - 드래그 영역 안에 존재하는 오브젝트들은 선택된 상태라는 것을 보여주기 위해 어둡게 표시.
  - 드래그 영역을 벗어나면 다시 원래 색으로 돌아올 수 있는 코드도 추가.
  - BlockHandler의 함수를 써서 영역 안에 존재하는 오브젝트들을 리스트로 관리.

```
BlockHandler blockHandler;

Unity 메시지 | 참조 0개
void Start()
{
    blockHandler = GameObject.Find("BlockHandler").GetComponent<BlockHandler>();
}

Unity 메시지 | 참조 0개
void OnTriggerEnter(Collider col)
{
    blockHandler.AddBlock(col.gameObject);
    col.gameObject.GetComponent<SpriteRenderer>().color = new Color(0.15f, 0.15f, 0.15f);
}

Unity 메시지 | 참조 0개
void OnTriggerExit(Collider col)
{
    blockHandler.RemoveBlock(col.gameObject);
    col.gameObject.GetComponent<SpriteRenderer>().color = new Color(0.8f,0.8f,0.8f,1);
}
```

```
참조 1개
public void AddBlock(GameObject block)
{
    blockList.Add(block);
}

참조 1개
public void RemoveBlock(GameObject block)
{
    blockList.Remove(block);
}
```

↑ : BlockHandler

프로젝트 수행 결과물

- **게임 로직 구현**
  - BlockHandler는 오브젝트들을 전반적으로 관리하는 새 클래스.
  - 스테이지들은 저장해뒀던 배열을 기반으로 오브젝트 프리팹을 생성.
  - 마우스 버튼이 올라갈 때마다 그 합이 10이 되는지 확인.
  - 오브젝트가 제거되면, 점수도 증가시키고 효과음도 재생.
  - 마우스 버튼에서 손을 떼었을 때는 드래그 영역이 제거되었을 때이므로 영역 안에 들어왔던 오브젝트들에 대한 리스트 리셋.

```
Unity 메시지 | 참조 0개
void Start()
{
    if (stage == 0)
    {
        // 블럭 랜덤 생성
        for (int i = 0; i < 170; i++)
        {
            var block = Instantiate(blockPrefab, new Vector2(0, 0), Quaternion.identity, GameObject.Find("BlockLayout").transform);
            block.transform.GetChild(0).gameObject.GetComponent<Text>().text = Random.Range(1, 10).ToString();
        }
    } else if (stage == 1)
    {
        for (int i = 0; i < 150; i++)
        {
            var block = Instantiate(blockPrefab, new Vector2(0, 0), Quaternion.identity, GameObject.Find("BlockLayout").transform);
            block.transform.GetChild(0).gameObject.GetComponent<Text>().text = StageOneNum[i].ToString();
        }
    }
    else if (stage == 2)
    {
        for (int i = 0; i < 150; i++)
        {
            var block = Instantiate(blockPrefab, new Vector2(0, 0), Quaternion.identity, GameObject.Find("BlockLayout").transform);
            block.transform.GetChild(0).gameObject.GetComponent<Text>().text = StageTwoNum[i].ToString();
        }
    }
}
Unity 메시지 | 참조 0개
void Update()
{
    if (Input.GetMouseButtonUp(0))
    {
        int sum = 0;
        foreach (GameObject block in blockList)
        {
            sum += int.Parse(block.transform.GetChild(0).gameObject.GetComponent<Text>().text);
            block.GetComponent<SpriteRenderer>().color = new Color(0.8f, 0.8f, 0.8f);
        }

        if (sum == 10)
        {
            foreach (GameObject block in blockList)
            {
                block.GetComponent<SpriteRenderer>().color = new Color(0, 0, 0, 0);
                block.transform.GetChild(0).gameObject.GetComponent<Text>().color = new Color(0, 0, 0, 0);
                block.GetComponent<BoxCollider>().isTrigger = false;
                Score.curScore++;
                SoundManager.Instance.SfxPlay(1,0.2f);
            }
        }
        blockList.Clear();
    }
}
```

프로젝트 수행 결과물

- **게임 로직 구현**
  - 시간 UI 추가.
  - 시간이 다 가면, 드래그 영역이 안 생기도록 EndGame함수를 불러옴.
  - 끝났을 때 Score에 있는 함수로 최고점을 저장하는데, 최고점이 아니더라도 함수 내에서 고려해줌.
  - BGM도 종료되며, 만약 목표 점수가 넘었다면 종료 시 뜨는 팝업창에 완료 문구가 추가되도록 함.
  - 챌린지 모드에는 목표점수가 없으므로 조건문에 추가.

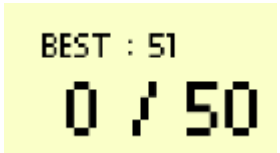

```
[SerializeField]
Slider timer;

[SerializeField]
GameObject gameEndPopup;
[SerializeField]
GameObject isSuccess;

public float gameTime;
public bool isEnd;
public static bool stopTimer;

Score score;

public int challengeMode = 0;
Unity 메시지 | 참조 0개
void Start()
{
    stopTimer = false;
    isEnd = false;
    timer.maxValue = gameTime;
    timer.value = gameTime;

    score = GameObject.Find("Score").GetComponent<Score>();
}
```

```
void Update()
{
    float time = gameTime - Time.timeSinceLevelLoad;

    if (time <= 0.1f)
    {
        timer.value = 0;
        if(isEnd == false)
        {
            EndGame();
        }
    }

    if(stopTimer==false)
    {
        Time.timeScale = 1;
        timer.value = time;
    }
    else
    {
        Time.timeScale = 0;
    }
}
```

```
void EndGame()
{
    stopTimer = true; //타이머 멈추기
    gameEndPopup.SetActive(true); //게임 끝 팝업
    if (challengeMode == 0)
    {
        if (score.success) { isSuccess.SetActive(true); } else { isSuccess.SetActive(false); }
    }
    MouseHandler.mouseActive = false; //마우스 드래그 못하게
    score.SaveMaxScore(); //끝났을 때 게임 저장
    SoundManager.Instance.SfxPlay(2, 0.2f); //효과음
    SoundManager.Instance.BgmStop();

    isEnd = true; //게임 끝
}
```

프로젝트 수행 결과물

- **게임 로직 구현**
  - 최대 점수가 목표 점수를 넘으면 그 스테이지는 성공한 스테이지라는 논리를 기반으로 Playerprefs에 배열 형태로 각 스테이지당 최고 점수 저장.
  - Index가 0일 때는 챌린지 모드의 최고점, 1부터는 각 스테이지들의 최고점을 저장.
  - 저장된 최고 점수는 각 스테이지 화면에서 볼 수 있음.
  - 최고 점수가 계속 저장되어 있는 상태이기 때문에 스테이지 클리어 유무를 이걸로 확인할 수 있음.



점수

```
[SerializeField]
Text curScoreText, maxScoreText;

[SerializeField]
Text gameEndScoreText;

[SerializeField]
Text goalScore;

public bool success = false;
public int stage = 0;
public int goal = 50;
public static int curScore = 0;
public static int[] maxScore = { 0, 0, 0, 0 };

Unity 메시지 | 참조 0개
void Start()
{
    curScore = 0;
    LoadMaxScore();
    if(stage != 0)
        goalScore.text = "/ "+goal.ToString();
}
void Update()
{
    curScoreText.text = curScore.ToString();
    maxScoreText.text = "BEST : " + maxScore[stage].ToString();
    gameEndScoreText.text = curScore.ToString();

    if (maxScore[stage] < curScore)
    {
        maxScore[stage] = curScore;
    }
    if(curScore >= goal)
    {
        success = true;
    }
}

참조 1개
public void LoadMaxScore()
{
    maxScore[stage] = PlayerPrefs.GetInt("MaxScore_Stage" + stage, 0);
}

참조 1개
public void SaveMaxScore()
{
    PlayerPrefs.SetInt("MaxScore_Stage" + stage, maxScore[stage]);
    PlayerPrefs.Save();
}
```

프로젝트 수행 결과물

- **게임 로직 구현**
  - 시간이 다 지나서 뜨는 팝업창과 별개로, Pause 버튼을 누르면 새로운 팝업창이 뜨며 이동 가능한 Scene으로 가는 버튼 존재.
  - 해당 코드는 일단 Pause 버튼과 Play 버튼의 작동 함수임.
  - 별도로 Sound를 껐다 키는 기능도 추가.
  - Sound가 켜져있는지 꺼져있는지에 따라 이미지를 변경하고, 버튼이 눌리면 현재의 sound 상태를 PlayerPrefs로 저장하여 소리 상태가 모든 씬에서 일정하도록 함.

```
[SerializeField]
GameObject pausePopup;

참조 0개
public void PauseButton()
{
    Timer.stopTimer = true;
    pausePopup.SetActive(true);
    MouseHandler.mouseActive = false;
    SoundManager.Instance.SfxPlay(0,0.2f);
    SoundManager.Instance.BgmPause();
}

참조 0개
public void PlayButton()
{
    Timer.stopTimer = false;
    pausePopup.SetActive(false);
    MouseHandler.mouseActive = true;
    SoundManager.Instance.SfxPlay(0,0.2f);
    SoundManager.Instance.BgmPlay();
}
```

Pause 버튼을 누르면 나오는 팝업창

```
[SerializeField]
Sprite soundOn, soundOff;

[SerializeField]
Image img;

Unity 메시지 | 참조 0개
public void Start()
{
    if(SoundManager.activeSound == 1)
    {
        img.sprite = soundOn;
    }
    else if(SoundManager.activeSound == 0)
    {
        img.sprite = soundOff;
    }
}

public void SoundButtonClick()
{
    if(SoundManager.activeSound == 1)
    {
        SoundManager.activeSound = 0;
        SoundManager.Instance.BgmPause();

        img.sprite = soundOff;

        PlayerPrefs.SetInt("ActiveSound", 0);
        PlayerPrefs.Save();
    }
    else if(SoundManager.activeSound == 0)
    {
        SoundManager.activeSound = 1;
        SoundManager.Instance.BgmPlay();

        img.sprite = soundOn;

        PlayerPrefs.SetInt("ActiveSound", 1);
        PlayerPrefs.Save();
    }
}
```

프로젝트 수행 결과물

- **게임 로직 구현**
  - BGM을 키거나 끄는 함수도 있고, 멈추는 함수도 있으며, 해당 함수들을 버튼 트리거에 의해 사용할 수 있음.
  - 효과음을 재생하는 함수도 있으며, 효과음마다 Index를 부여하여 상황에 알맞은 효과음을 재생할 수 있도록 함.

```
[SerializeField]
AudioSource bgmAudioSource;

[SerializeField]
AudioSource sfxAudioSource;

[SerializeField]
AudioClip bgm;
[SerializeField]
AudioClip[] sfx;

[SerializeField]
float bgmVolume;
```

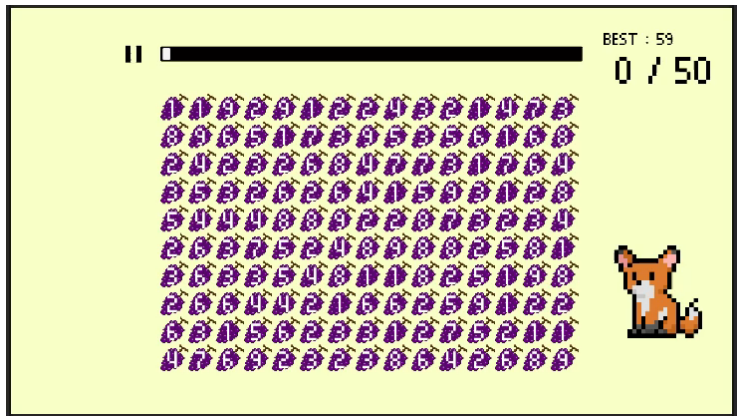
```
public static int activeSound=1;

private static SoundManager instance;
참조 20개
public static SoundManager Instance
{
    get
    {
        if (instance == null) instance = new SoundManager();
        return instance;
    }
}
```

```
void Awake()
{
    if(instance == null)
    {
        instance = this;
    }
    else if(instance != this)
    {
        Destroy(gameObject);
    }
    DontDestroyOnLoad(gameObject);

    if (PlayerPrefs.HasKey("ActiveSound"))
    {
        activeSound = PlayerPrefs.GetInt("ActiveSound");
    }
}

참조 6개
public void BgmPlay()
{
    if(activeSound == 1)
    {
        bgmAudioSource.clip = bgm;
        bgmAudioSource.loop = true;
        bgmAudioSource.volume = bgmVolume;
        bgmAudioSource.Play();
    }
}
```

```
public void BgmStop()
{
    bgmAudioSource.Stop();
}

참조 2개
public void BgmPause()
{
    bgmAudioSource.Pause();
}

참조 10개
public void SfxPlay(int index, float volume) // 0은 클릭, 1은 블록 터뜨리기, 2는 게임 끝
{
    if(activeSound == 1)
    {
        sfxAudioSource.volume = volume;
        sfxAudioSource.PlayOneShot(sfx[index]);
    }
}
```

프로젝트 수행 결과물

- **게임 로직 구현**
  - Scene을 이동하는 모든 함수들은 SceneHandler 스크립트에 모아둠.
  - 스테이지 선택화면, 각 스테이지들, 챌린지 화면, 로비로 향하는 함수들이 있으며, Stage3과 Stage4는 예비로 제작해 둠.
  - 게임을 나가는 함수도 존재.




```
public void ChallengeStartButton()
{
    SceneManager.LoadScene("Challenge");
    SoundManager.Instance.SfxPlay(0, 0.2f);
    SoundManager.Instance.BgmPlay();
}

참조 0개
public void LobbyButton()
{
    SceneManager.LoadScene("Lobby");
    SoundManager.Instance.SfxPlay(0, 0.2f);
    SoundManager.Instance.BgmStop();
}

참조 0개
public void ExitButton()
{
    SoundManager.Instance.SfxPlay(0, 0.2f);
    Application.Quit();
}
```

```
public void StartButton()
{
    SoundManager.Instance.SfxPlay(0, 0.2f);
    SceneManager.LoadScene("StageSelect");

    SoundManager.Instance.BgmPlay();

}
```

```
public void StageStartButton()
{
    if (StageHandler.currentTargetIndex == 0)
    {
        SceneManager.LoadScene("Play 1");
        SoundManager.Instance.SfxPlay(0, 0.2f);
        SoundManager.Instance.BgmPlay();
    }
    else if (StageHandler.currentTargetIndex == 1)
    {
        SceneManager.LoadScene("Play 2");
        SoundManager.Instance.SfxPlay(0, 0.2f);
        SoundManager.Instance.BgmPlay();
    }
    else if (StageHandler.currentTargetIndex == 2)
    {
        // 후에 추가 가능
        //SceneManager.LoadScene("Play 3");
        //SoundManager.Instance.SfxPlay(0, 0.2f);
        //SoundManager.Instance.BgmPlay();
    }
    else if (StageHandler.currentTargetIndex == 3)
    {
        // 후에 추가 가능
        //SceneManager.LoadScene("Play 4");
        //SoundManager.Instance.SfxPlay(0, 0.2f);
        //SoundManager.Instance.BgmPlay();
    }
```

프로젝트 수행 결과물

- **스테이지 선택 화면 구현**
  - 각 스테이지들을 하나로 묶어서 그룹의 위치를 통해 이동 구현.
  - Lerp를 통해 부드러운 이동을 유도.


```
[SerializeField]
public GameObject targetObject; // 이동할 오브젝트
public Vector3[] targetPositions = { new Vector3(0, 0, 0), new Vector3(-10, 0, 0), new Vector3(-20, 0, 0), new Vector3(-30, 0, 0) }; // 목표 위치
public float moveSpeed = 3.0f; // 이동 속도

public static int currentTargetIndex = 0;
private bool shouldMove = false; // 이동 여부

Unity 메시지 | 참조 0개
void OnEnable()
{
    currentTargetIndex = 0;
    shouldMove = true;
}
Unity 메시지 | 참조 0개
void Update()
{
    if (shouldMove)
    {
        Vector3 targetPosition = targetPositions[currentTargetIndex];
        targetObject.transform.position = Vector3.Lerp(targetObject.transform.position, targetPosition, moveSpeed * 0.01f);

        if (Vector3.Distance(targetObject.transform.position, targetPosition) < 0.01f)
        {
            targetObject.transform.position = targetPosition;
            shouldMove = false;
        }
    }
}

참조 0개
public void OnLeftButtonClicked()
{
    if (currentTargetIndex > 0)
    {
        currentTargetIndex = currentTargetIndex - 1;
        shouldMove = true;
    }

}
참조 0개
public void OnRightButtonClicked()
{
    if (currentTargetIndex < targetPositions.Length - 1)
    {
        currentTargetIndex += 1;
        shouldMove = true;
    }
}
```

프로젝트 수행 결과물

- **스테이지 선택 화면 구현**
  - 이미지 혹은 스프라이트로 되어있는 동물이나 음식이 스테이지를 클리어하면 색을 되찾을 수 있도록 하는 코드.

```
public int stage;
private Image image;
private SpriteRenderer spriteRenderer;

Unity 메시지 | 참조 0개
void Update()
{
    Image image = GetComponent<Image>();
    spriteRenderer = GetComponent<SpriteRenderer>();
    int score_ = PlayerPrefs.GetInt("MaxScore_Stage" + stage, 0);
    if (score_ >= 50)
    {
        if(image != null)
            image.color = Color.white;
        else if(spriteRenderer != null)
            spriteRenderer.color = Color.white;
    }
}
```

계획 대비 결과물 진척도

- **마우스 드래그 영역 및 그에 따른 오브젝트 상호작용 가시화**
- **스테이지 2개 제작 완료**
    - 로비 화면이나 스테이지 선택 화면이 허전해서 두 개의 껍데기 스테이지도 제작 완료
- **챌린지 모드 제작 완료**
- **스테이지 클리어 정보를 불러와, 로비 및 스테이지 선택 화면에서 동물 수집 확인**
- **동적인 스테이지 선택 화면 제작 완료**
- **잠깐 게임 화면을 멈출 수 있는 팝업창 제작**
    - 멈춘 화면이 투명하면 게임에 남용될 수 있으므로 불투명하게 수정 완료
- **효과음 및 BGM 추가**

게임 플레이 동영상