```
UNITY_EDITOR
//  파일네임으로 가이드인지 파악해서 제거해주는 기능
[MenuItem("Assets/SubFunctions/DeleteGuide", priority = 1, validate = false)]
static void DelGuide()
{
    var selectedAssets = Selection.GetFiltered(typeof(UnityEngine.Object), SelectionMode.Assets);
    foreach (var asset in selectedAssets)
    {
        bool check = asset.name.Contains("0_Guide")||asset.name.Contains("0_guide");

        if (check)
        {
            string assetPath = AssetDatabase.GetAssetPath(asset);
            File.Delete(assetPath);
        }
    }
}

//  [폴더명+파일명]으로 파일명을 변경합니다.일반적으로 아바타 리소스에 적용합니다.
[MenuItem("Assets/SubFunctions/RenameByFolderNameForAvartar", priority = 2, validate = false)]
static void RenameTextureByFolderName2()
{
    var selectedAssets = Selection.GetFiltered(typeof(UnityEngine.Object), SelectionMode.Assets);

    foreach (var asset in selectedAssets)
    {
        string assetPath = AssetDatabase.GetAssetPath(asset);
        string extension = Path.GetExtension(assetPath);

        string basePath = Path.GetDirectoryName(assetPath);
        var splits = basePath.Split('\\');
        string folderName = splits[splits.Length - 1];
        splits = folderName.Split('_');
        string folderNum = splits[splits.Length - 1];

        string assetName = folderNum + Path.GetFileNameWithoutExtension(assetPath) + extension;
        string newPath = Path.Combine(basePath, assetName);
        AssetDatabase.RenameAsset(assetPath, assetName);
    }
    AssetDatabase.SaveAssets();
}
#if
```

코드설명

DelGuide는 드래그 한 파일들을 가져온 후 본인이 지정한 이름이 있을 경우 해당 파일을 삭제합니다.

RenameTextureByFolderName은 드래그 한 파일을 가져와 그 경로를 string으로 반환 후 경로 폴더 이름의 마지막 string을 split으로 잘라 이름을 변경시킵니다.

```
Properties
{
    _MainTex ("Texture", 2D) = "white" {}
    _Speed("speed",Float) = 5
    _Amp("Amplitude",Float) = 0.05

        //Usedfor UI.Mask
        _StencilComp("Stencil Comparison",Float) = 8
        _Stencil("Stencil ID",Float) = 0
        _StencilOp("Stencil Operation",Float) = 0
        _StencilWriteMask("Stencil Write Mask",Float)=255
        _StencilReadMask("Stencil Read Mask",Float) = 255
        _ColorMask("Color Mask",Float)=15

}
```

```
        struct v2f
        {
            float2 uv : TEXCOORD0;
            float4 vertex : SV_POSITION;
            float4 color : COLOR;
        };

        v2f vert (appdata v)
        {
            v2f o;
            o.vertex = UnityObjectToClipPos(v.vertex);
            o.uv = v.uv;
            o.color = v.color;
            return o;
        }

        sampler2D _MainTex;
        float _Speed;
        float _How;
        float _Amp;

        fixed4 frag (v2f i) : SV_Target
        {
            float temp = i.uv.y*i.uv.y* sin(_Time.y*_Speed)*_Amp;

            fixed4 col = tex2D(_MainTex, float2(i.uv.x+temp,i.uv.y));

            col.rgb*=i.color.rgb;
            col.a*=i.color.a;
            col.rgb *= col.a;
            return col;
        }
        ENDCG
        Cull Off
    }
```

코드설명

유니티에서 제공하는 Shader로
이미지를 옆으로 움직이게 하여
바람에 휘날리듯 보이게 하는 코드
입니다.

```
Shader "Custom/water_shader"
{
    Properties
    {
        _MainTex ("Texture", 2D) = "white" {}
        _Speed("speed",Float) = 5
        _How("HowManyWave",Float) = 5
        _Amp("Amplitude",Float) = 0.05
    }
```

```
    v2f vert (appdata v)
    {
        v2f o;
        o.vertex = UnityObjectToClipPos(v.vertex);
        o.uv = v.uv;
        o.color = v.color;
        return o;
    }

    sampler2D _MainTex;
    float _Speed;
    float _How;
    float _Amp;


    fixed4 frag (v2f i) : SV_Target
    {
        float temp = sin(_Time.y*_Speed + i.uv.x*_How);


        fixed4 col = tex2D(_MainTex, float2(i.uv.x,i.uv.y+_Amp*temp));
        col.rgb*=i.color.rgb;
        col.a*=i.color.a;
        col.rgb *= col.a;
        return col;
    }
    ENDCG
    Cull Off
}
```

코드설명

유니티에서 제공하는 Shader로
이미지를 옆으로 sin그래프처럼
움직이게 해서 물이 두둥실 떠다니듯
연출하게 합니다.

```
Shader "Custom/gradationY"
{
    Properties
    {
        _MainTex("Texture", 2D) = "white" {}
        _SmokeTime("smoketime",Float) = 0
        _UpandDown("UpAndDown",int) = 1
    }
fixed4 color;
color = tex2D(_MainTex, float2(i.uv.x,i.uv.y));
color *= i.color;
if(_UpandDown>0.5f)
{
    if (_SmokeTime < i.uv.y )
    {
        color.a *= (_SmokeTime)*-1 + (i.uv.y);
    }
    else
    {
        color.a *= 0;
    }
}
else
{
    if (_SmokeTime > i.uv.y )
    {
        color.a *= _SmokeTime - i.uv.y;
    }
    else
    {
        color.a *= 0;
    }
}
if(color.a>1)
    color.a=1;


color.rgb *= color.a;

return color;

}
ENDCG
Cull Off
```

코드설명

유니티에서 제공하는 Shader로 이미지는 전체적으로 투명도로 사라져 만든 Shader로 이미지를 위나 아래부터 투명도로 사라지게 하여 연출에 자연스러움을 줍니다.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
참조 1개
public abstract class GetInstance<T> : MonoBehaviour where T : GetInstance<T>
{
    protected static T instance;
    // Start is called before the first frame update
    참조 0개
    public static T Instance()
    {
        if (instance == null)
            instance = FindObjectOfType<T>();
        return instance;
    }
    참조 0개
    protected virtual void Awake()
    {

        if (instance == null)
        {
            instance = this as T;
        }
        else
        {
            if (instance != this)
            {
                Destroy(gameObject);
            }
        }
    }
public class keypad : GetInstance<keypad>
{
    public int num;
public class NewBehaviourScript : MonoBehaviour
{
    // Start is called before the first frame update
    Unity 메시지|참조 0개
    void Start()
    {
        keypad.Instance().num = 0;
```

코드설명

싱글톤을 매번 해주는 번거로움을 없애기 위해 Getinstance를 사용하여 스크립트를 제네릭으로 넣고, 싱글톤을 다른 스크립트에서 쉽게 사용 가능하게 합니다.

```
public class Delegate_Keypad_WS : MonoBehaviour
{
    private static Delegate_Keypad_WS instance;
    public static Delegate_Keypad_WS Instance
    {
        get { return instance; }
    }

    private void Awake()
    {
        if (instance)
        {
            Destroy(gameObject);
            return;
        }
        instance = this;
    }

    public Keypad_CWS keypad_ws;
    public bool pointcheck;

    public delegate void Delegate_void();
    public delegate void Delegate_void_input(string a);
    public delegate bool Delegate_bool();

    Delegate_void_input D_EnterKeyEvent, D_InputKeyEvent;
    Delegate_bool D_AnswerCheck;
```

```
public void SetDelegate(KeyPadUser padUser, bool check_multi = true, int num_Natrueline = 2, int num_Underpoint = 3)
{
    D_EnterKeyEvent = padUser.EnterKeyEvent;
    D_AnswerCheck = padUser.GetAnswerCheck;
    D_InputKeyEvent = padUser.InputKeyEvent;

    keypad_ws.multi_check = check_multi;
    keypad_ws.num_N_line = num_Natrueline;
    keypad_ws.num_Point_line = num_Underpoint;

}
```

코드설명

앱에 키패드 기능을 매번 가져오기 번거로워 Delegate로 만들어 사용하기 쉽게 만드는 코드입니다.

```
private void Reset()
{
    lineRenderer = GetComponent<LineRenderer>();
    lineRenderer.startColor = Color.white;
    lineRenderer.endColor = Color.white;

}

private void Awake()
{
    lineRenderer.useWorldSpace = false;
    Camera camera = Camera.main;
    Vector3 Temp = Camera.main.ViewportToWorldPoint(Vector3.one) - Camera.main.ViewportToWorldPoint(Vector3.zero);
    if (Temp.x > Temp.y / 9 * 16)
        Temp.x = Temp.y / 9 * 16;
    else
        Temp.y = Temp.x * 9 / 16;
    screenx = Temp.x;
    screeny = Temp.y;

    float SizeY = (screeny / 2) / Camera.main.orthographicSize;
    ropeWidth *= SizeY;
    if (t != null)
    {
        Transform[] g = t.GetComponentsInChildren<Transform>();
        foreach (Transform ga in g)
        {
            if (ga != t)
                moveobejct.Add(ga);
        }
    segmentCount = moveobejct.Count;
    }
    startTransform.GetComponent<RectTransform>().anchoredPosition3D = new Vector3(startTransform.GetComponent<RectTransform>().anchoredPosition3D.x,
        startTransform.GetComponent<RectTransform>().anchoredPosition3D.y);
    //segmentLength *= SizeY;
    //시작할때 점 위치지정
    Vector2 segmentPos = new Vector2(startTransform.GetComponent<RectTransform>().anchoredPosition3D.x, startTransform.GetComponent<RectTransform>().anchoredPosition3D.y);
    for(int i=0;i<segmentCount;i++)
    {
        segments.Add(new Segment(segmentPos));
        segmentPos.y -= segmentLength; //계속 아래로
    }
}
```

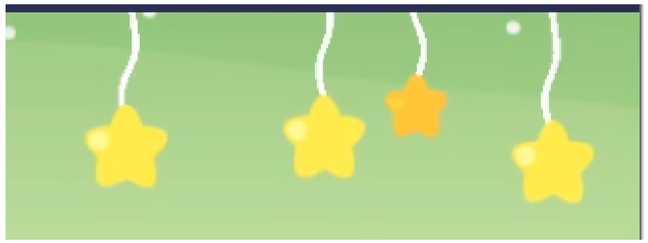
```
private void FixedUpdate()
{
    UpdateSegemts();
    for(int i=0;i<constrairtLoop;i++)
    {
        ApplyConstraint();
    }
    DrawRope();
}

/// <summary>
/// 라인렌더러로 점 그려줌
/// </summary>
private void DrawRope()
{
    if (t == null)
    {
        lineRenderer.startWidth = ropeWidth;
        lineRenderer.endWidth = ropeWidth;
    }
    Vector3[] segmentPositions = new Vector3[segments.Count];
    for (int i=0;i<segments.Count;i++)
    {
        //segmentPositions[i] = segments[i].position;
        segmentPositions[i] = new Vector3(segments[i].position.x, segments[i].position.y, 0);
        if (t != null)
        {
            moveobejct[i].transform.position = segments[i].position;
        }
    }
    Vector3 imageLength = new Vector3(segments[segments.Count - 1].position.x, segments[segments.Count - 1].position.y, 0) - new Vector3(0, ImageLength, 0);
    if (image!=null)
    image.transform.GetComponent<RectTransform>().anchoredPosition3D= imageLength;

    if (t == null)
    {
        lineRenderer.positionCount = segmentPositions.Length;
        lineRenderer.SetPositions(segmentPositions);
    }
}
```

```
private void UpdateSegemts()
{
    for(int i=0;i<segments.Count;i++)
    {
        segments[i].velocity = segments[i].position - segments[i].previousPos;
        segments[i].previousPos = segments[i].position;
        segments[i].position += gravity * Time.fixedDeltaTime * Time.fixedDeltaTime;
        segments[i].position += segments[i].velocity;
    }
}

private void ApplyConstraint()
{
    segments[0].position = startTransform.GetComponent<RectTransform>().anchoredPosition3D;
    for(int i=0;i<segments.Count-1;i++)
    {
        float distance = (segments[i].position - segments[i + 1].position).magnitude;
        float difference = segmentLength - distance;
        Vector2 dir =(segments[i + 1].position - segments[i].position).normalized;

        Vector2 movement = dir * difference;
        if (i == 0)
            segments[i + 1].position += movement;
        else
        {
            segments[i].position -= movement*0.5f;
            segments[i + 1].position += movement * 0.5f;
        }
    }
}

public class Segment
{
    public Vector2 previousPos;
    public Vector2 position;
    public Vector2 velocity;




    public Segment(Vector2 _position)
    {
        previousPos = _position;
        position = _position;
        velocity = Vector2.zero;
    }
}
```

코드설명

유니티에서 제공하는 LineRenderer로 거미줄 같이 흔들리는 선을 만드는 기능입니다.
선을 원하는 만큼 만든 후
선 하나하나에 시작 위치를
받은 후에 중력 값만큼 아래로 쭉 내리며
물리 처리를 하기 위해 Fixed를 써서 구현했습니다.

예시이미지



```
public static void NumAdd(this List<int> _a, int _num)
{
    int rand = Random.Range(0, _num);
    for (int i = 0; i < _num;)
    {
        if (_a.Contains(rand))
        {
            rand = Random.Range(0, _num);
            continue;
        }
        else
        {
            _a.Add(rand);
            i++;
        }
    }
}
```

코드설명

_a의 리스트 안에 중복되지 않는 랜덤을 집어넣습니다.

```
public static IEnumerator SetScaleByAnimationCurve1(this Transform transform, AnimationCurve curve, float speed = 1, bool isSetLastValue = false)
{
    float targetTime = curve.GetPlayTime();
    float curTime = 0;
    Vector3 originScale = transform.localScale;

    while (curTime <= targetTime)
    {
        float scalar = curve.Evaluate(curTime);
        float scale = scalar * originScale.x;
        float scale1 = scalar * originScale.y;

        transform.localScale = new Vector2(scale, scale1);

        yield return null;

        curTime += Time.deltaTime * speed;
    }

    transform.localScale = isSetLastValue ? Vector3.one * curve.GetLastValue() : originScale;
}
```

코드설명

transform에 애니메이션커브를 이용하여 그래프 크기만큼 커졌다 작아지게 하는 팝핀 기능

```
/// <summary>
/// 틀렸을때 상자흔들림
/// </summary>
/// <param name="target">어떤게흔들릴지</param>
/// <param name="time">흔들릴시간</param>
/// <param name="power">크기</param>
/// <returns></returns>
public static IEnumerator ShakeObjectByAnchored3D(this RectTransform target, float time = 1f, float power = 20f, float width = 0.5f)
{
    Vector3 origin = target.anchoredPosition3D;

    float curWidth = width;

    float timer = 0f;

    float curPower = power;

    var wait = new WaitForSeconds(0.01f);

    while (timer < time)
    {
        yield return wait;

        float posX = Random.Range(-curWidth, curWidth) * curPower;
        float posY = Random.Range(-curWidth, curWidth) * curPower;

        target.anchoredPosition3D = origin + new Vector3(posX, 0, 0);

        timer += Time.deltaTime * 2f;

        if (curPower > 0)
            curPower -= curWidth;

    }

    target.anchoredPosition3D = origin;

}
```

코드설명

target을 랜덤하게 왼쪽 오른쪽으로 빠르게 움직여 틀렸을 때 흔들림을 구현

```
public static IEnumerator AnimationEnd(this Animation _A, string s = null)
{
    if (s != null)
    {
        Animation _a = _A;
        AnimationClip c = null;
        foreach (AnimationState state in _a)
        {
            if (state.name == s)
            {
                c = state.clip;
            }
        }
        _A.Play(s);
        yield return new WaitForSeconds(c.length);
    }
    else
    {
        _A.Play();
        yield return new WaitForSeconds(_A.clip.length);
    }
}
```

코드설명

애니메이션의 클립을 가져와 길이만큼
기다려주는 코루틴

```
public static IEnumerator AnimationEnd(this Animator _A, string s, float f = 1f)
{
    yield return new WaitUntil(() => _A.GetCurrentAnimatorStateInfo(0).IsName(s) && _A.GetCurrentAnimatorStateInfo(0).normalizedTime >= f);
}
```

코드설명

애니메이터의 이름과 시간을 체크해
끝나는 지점을 체크해주는 함수

```
/// <summary>
/// 정해진시간동안 목표물로 이동
/// </summary>
/// <param name="transform">움직일 오브젝트</param>
/// <param name="p">이동할위치</param>
/// <param name="timeToMove">몇초동안갈지</param>
/// <returns></returns>
public static IEnumerator MoveToVector(this Transform transform, Vector3 p, float timeToMove = 1)
{
    var currentPos = transform.GetComponent<RectTransform>().anchoredPosition3D;
    var t = 0f;
    RectTransform rect = transform.GetComponent<RectTransform>();
    while (t < 1)
    {
        rect.anchoredPosition3D = Vector3.Lerp(currentPos, p, t);
        yield return null;
        t += Time.deltaTime / timeToMove;
    }
    rect.anchoredPosition3D = p;
}
/// <summary>
/// 정해진시간동안 목표물로 이동
/// </summary>
/// <param name="transform">움직일 오브젝트</param>
/// <param name="p">이동할위치</param>
/// <param name="timeToMove">몇초동안갈지</param>
/// <returns></returns>
public static IEnumerator MoveToVector_Curve(this Transform transform, AnimationCurve curve, Vector3 p, float timeToMove = 1)
{
    var currentPos = transform.GetComponent<RectTransform>().anchoredPosition3D;
    var t = 0f;
    RectTransform rect = transform.GetComponent<RectTransform>();
    while (t < 1)
    {
        float x = curve.Evaluate(t) * p.x;
        float y = curve.Evaluate(t) * p.y;
        rect.anchoredPosition3D = Vector3.Lerp(currentPos, new Vector3(x, y), t);
        yield return null;
        t += Time.deltaTime / timeToMove;
    }
    rect.anchoredPosition3D = p;
}
```

코드설명

오브젝트를 원하는 시간만큼
지정한 좌표까지 Lerp로 이동하게
만드는 코드

애니메이션커브를 이용하여 들쑥날쑥
하게 움직이는 것을 가능하게 만드는
코드

회사 스테이지와 밀접한 코드를 제외하고 자주 쓰는 코드들을 올려놨습니다.
자세한 내용은 구글 드라이브 참조 부탁 드립니다.
https://drive.google.com/drive/folders/1Qmg6ZL3X6c22B-GdtT1pdAsVmjhpadow?usp=sharing

긴 글 읽어주셔서 감사합니다.