**베지어 곡선(Bezier Curve)**

시작 좌표와 목표 좌표를 입력 받아 중간위치의 임의의 2개의 점을 생성하여 오브젝트를 목표 좌표로 부드럽게 이동시켜준다.라인, 도표, 그래프이(가) 표시된 사진

자동 생성된 설명

예시) 카이사의 Q스킬 이케시안 폭우

PC 게임, 전략 비디오 게임, 비디오 게임 소프트웨어, 어드벤처 게임이(가) 표시된 사진

자동 생성된 설명

Code

public class BezierCurves : MonoBehaviour

{

Vector2[] m\_points = new Vector2[4]; //꼭지점 좌표배열

private float m\_timerMax = 0; //최대시간

private float m\_timerCurrent = 0; //현재시간

private float m\_speed; //이동속도

public GameObject hiteffect; //이펙트

//시작 좌표, 목표 좌표, 이동속도, 꺽는정도를 입력받아 세팅해주는 함수

public void Init(Transform \_startTr, Transform \_endTr, float \_speed, float \_newPointDistanceFromStartTr, float \_newPointDistanceFromEndTr)

{

m\_speed = \_speed;

// 끝에 도착할 시간을 랜덤으로 준다

m\_timerMax = Random.Range(0.8f, 1.0f);

// 시작 지점

m\_points[0] = \_startTr.position;

// 시작 지점을 기준으로 랜덤 포인트 지정

m\_points[1] = \_startTr.position +

(\_newPointDistanceFromStartTr \* Random.Range(-1.0f, 1.0f) \* \_startTr.right) +

// X (좌, 우 전체)

(\_newPointDistanceFromStartTr \* Random.Range(-0.15f, 1.0f) \* \_startTr.up);

// Y (아래쪽 조금, 위쪽 전체)

// 도착 지점을 기준으로 랜덤 포인트 지정

m\_points[2] = \_endTr.position +

(\_newPointDistanceFromEndTr \* Random.Range(-1.0f, 1.0f) \* \_endTr.right) +

// X (좌, 우 전체)

(\_newPointDistanceFromEndTr \* Random.Range(-1.0f, 1.0f) \* \_endTr.up);

// Y (위, 아래 전체)

// 도착 지점

m\_points[3] = \_endTr.position;

transform.position = \_startTr.position;

}

void Update()

{

//타이머의 시간이 끝나면 종료

if (m\_timerCurrent > m\_timerMax)

{

return;

}

// 경과 시간 계산하기

m\_timerCurrent += Time.deltaTime \* m\_speed;

// 베지어 곡선으로 X,Y 좌표 얻기

transform.position = new Vector2(

CubicBezierCurve(m\_points[0].x, m\_points[1].x, m\_points[2].x, m\_points[3].x),

CubicBezierCurve(m\_points[0].y, m\_points[1].y, m\_points[2].y, m\_points[3].y)

);

}

// 콜라이더 충돌 시 데미지

private void OnCollisionEnter2D(Collision2D collision)

{

if(collision.transform.CompareTag("BossMonster") || collision.transform.CompareTag("Monster"))

{

collision.gameObject.GetComponent<MonsterManager>().UnderAttack(10, hiteffect);

}

}

/// <summary>

/// 3차 베지어 곡선

/// </summary>

/// <param name="a">시작 위치</param>

/// <param name="b">시작 위치에서 얼마나 꺾일 지 정하는 위치</param>

/// <param name="c">도착 위치에서 얼마나 꺾일 지 정하는 위치</param>

/// <param name="d">도착 위치</param>

/// <returns></returns>

private float CubicBezierCurve(float a, float b, float c, float d)

{

// (0~1)의 값에 따라 베지어 곡선 값을 구하기 때문에, 비율에 따른 시간을 구하기

float t = m\_timerCurrent / m\_timerMax; // (현재 경과 시간 / 최대 시간)

// 방정식

/\*

return Mathf.Pow((1 - t), 3) \* a

+ Mathf.Pow((1 - t), 2) \* 3 \* t \* b

+ Mathf.Pow(t, 2) \* 3 \* (1 - t) \* c

+ Mathf.Pow(t, 3) \* d;

\*/

float ab = Mathf.Lerp(a, b, t); //a와 b사이의 좌표 구하기

float bc = Mathf.Lerp(b, c, t); //b와 c사이의 좌표 구하기

float cd = Mathf.Lerp(c, d, t); //d와 d사이의 좌표 구하기

float abbc = Mathf.Lerp(ab, bc, t); //ab와 bc사이의 좌표 구하기

float bccd = Mathf.Lerp(bc, cd, t); //bc와 cd사이의 좌표 구하기

return Mathf.Lerp(abbc, bccd, t);

}

}

결과 화면

