[ 게임 엔진 응용프로그래밍 ]

**상용 게임 엔진을 사용한 게임 프로그램**

인하대학교 미래인재개발원 문화콘텐츠프로그래밍

김예슬

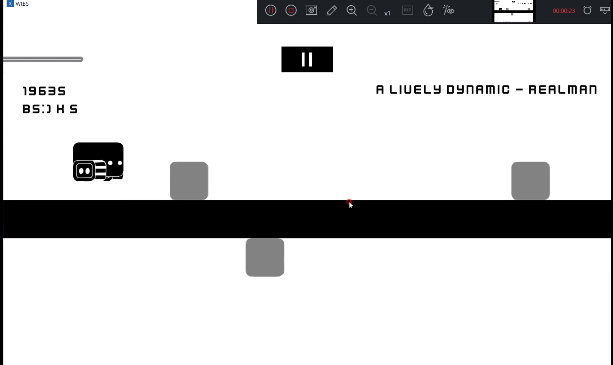
**1. 게임 설계**

- 플레이어는 자동으로 이동하며, 점프와 화면 삭제 기능을 사용할 수 있다.

- 장애물과의 충돌이 체크되어야 하며, 충돌 시 사망상태로 변경해서 체크 포인트부터 시작되어야

한다.

**2. 스크린샷**



**3. 소스코드**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement; // Scene change

using UnityEngine.EventSystems;

public class Player : MonoBehaviour

{

public static Rigidbody rigidBody; // 물리기능 이용

public static SpriteRenderer spriteRenderer; // filp 기능 이용

public static bool isGround; // 바닥 체크

public static bool playerPos; // 플레이어 위치

public static float playerMoveSpeed; // 이동속도 변수

public static float \_jumpPower; // 점프 힘 변수

public static int \_gravityForce; // 중력변수

public static int \_setPos; // 플레이어 위치

public static bool \_isDead; // 생사여부

public static int playerHitCount; // 충돌횟수

public static bool isPlayerCheckPoint; // 플레이어 체크포인트 지나침 확인

void Start()

{

Physics.gravity = Vector3.down \* 100; // 시작시 공중뜨는거 방지

rigidBody = GetComponent<Rigidbody>(); // Rigidbody 컴포넌트를 받아옴

spriteRenderer = GetComponent<SpriteRenderer>(); // SpriteRenderer 컴포넌트 받아옴

playerPos = true; // 위에 위치

isGround = true; // 점프 가능

playerHitCount = 0; // 부딪힘 초기화

isPlayerCheckPoint = false; // 체크포인트를 지나지 x

playerMoveSpeed = DataManager.moveSpeed;

\_jumpPower = DataManager.jumpPower;

\_gravityForce = DataManager.gravityForce;

\_setPos = DataManager.setPos;

\_isDead = DataManager.isDead;

// 변수 값 초기화

} // Start()

void Update()

{

if (!Game.isPause && !EndEffect.isEndEffect)

{

AutoMove();

InputMoveForPC();

}

} // Update()

public void InputMoveForPC() // 플레이어 입력 이동ver. PC

{

if (Input.GetKeyDown(KeyCode.Space)) // 점프

{

PlayerJump();

}

if (Input.GetKeyDown(KeyCode.UpArrow) || Input.GetKeyDown(KeyCode.DownArrow)) // 위치 변환

{

CheckPlayerPos();

}

if (Input.GetKeyDown(KeyCode.LeftArrow) || Input.GetKeyDown(KeyCode.RightArrow)) // 화면 삭제

{

Game.BlueScreenOn();

}

} // InputMove()

public static Vector3 lastPlayerPos; // Pause 상태일 때 플레이어가 튀어나가는 것 방지

public void AutoMove() // 플레이어 자동 이동

{

lastPlayerPos = transform.position; // 플레이어 위치 저장

transform.Translate(playerMoveSpeed \* Time.deltaTime, 0f, 0f);

} // AutoMove()

public static bool rockChangePos = false; // 점프 중 위치 변환 x

public static void PlayerJump() // 플레이어 점프

{

if (isGround) // 땅에 위치할 경우

{

if (playerPos)

{

rigidBody.AddForce(Vector3.up \* \_jumpPower, ForceMode.Impulse);

}

else

{

rigidBody.AddForce(Vector3.down \* \_jumpPower, ForceMode.Impulse);

}

}

rockChangePos = false;

} // PlayerJump()

public static void CheckPlayerPos() // 플레이어 위치 판별

{

if (rockChangePos == true && playerPos == true) // 위에서 아래로

{

playerPos = false;

spriteRenderer.flipY = true;

Physics.gravity = Vector3.up \* \_gravityForce;

GameObject.FindGameObjectWithTag("Player").transform.Translate(Vector3.down \* \_setPos);

}

else if (rockChangePos == true && playerPos == false) // 아래에서 위로

{

playerPos = true;

spriteRenderer.flipY = false;

Physics.gravity = Vector3.down \* \_gravityForce;

GameObject.FindGameObjectWithTag("Player").transform.Translate(Vector3.up \* \_setPos);

}

} // CheckPlayerPos()

/\*

물체 충돌 관련 함수들

- 점프 횟수 설정(1번만 가능)

- 위치 변경 설정(점프 중 불가능)

- 장애물과 부딪혔을 경우

\*/

void OnCollisionEnter(Collision collision) // 처음 부딪혔을 경우

{

SetCollisionVal(collision, 1);

} // OnCollisionEnter(Collision collision)

void OnCollisionStay(Collision collision) // 부딪힌 상태 유지

{

SetCollisionVal(collision, 1);

} // OnCollisionStay(Collision collision)

void OnCollisionExit(Collision collision) // 콜라이더 떨어져 있을 때

{

SetCollisionVal(collision, -1);

} // OnCollisionExit(Collision collision)

void SetCollisionVal(Collision collision, int check) // 충돌관련 변수 set

{

if (collision.transform.tag == "Ground" && check == 1)

{

isGround = true; //바닥과 맞닿아 있음(점프 가능)

rockChangePos = true; // 위치변경 가능

}

else

{

isGround = false; //바닥과 맞닿아 있지 않음(점프 불가능)

rockChangePos = false; // 위치변경 불가능

}

} // SetCollisionVal()

/\*

isTriger를 이용, hitCount 체크

\*/

public static bool isNoHit = false; // 부딪침 체크

public static bool isHiting = false;

private void OnTriggerEnter(Collider other) // 장애물과 부딪혔을 경우

{

if ( other.transform.tag == "Obstacle" && !isNoHit )

{

isNoHit = true;

isHiting = true;

StartCoroutine("NoHitTime");

playerHitCount++; // 부딪힐 경우 증가

GetComponent<PlayerPos>().PlayerCheckPoint();

} // 충돌 체크 시 카운트 증가 -> Game.cs에서 점수 차감

} // OnTriggerEnter(Collider other)

IEnumerator NoHitTime() // 무적시간용 코루틴 함수

{

Player.rockChangePos = false;

Player.playerPos = true;

Player.spriteRenderer.flipY = false;

Physics.gravity = Vector3.down \* Player.\_gravityForce;

GameObject.FindGameObjectWithTag("Player").transform.Translate(Vector3.up \* Player.\_setPos);

int countTime = 0;

while(countTime<10)

{

if (countTime%2==0)

{

spriteRenderer.color = new Color32(255, 255, 255, 90);

}

else

{

spriteRenderer.color = new Color32(255, 255, 255, 180);

}

// 무적인 동안 깜빡임 효과

yield return new WaitForSeconds(0.15f);

countTime++;

}

spriteRenderer.color = new Color32(255, 255, 255, 255);

isNoHit = false;

yield return null;

} // IEnumerator NoHitting()

} // Player Class