캡스톤디자인(종합설계) 결과보고서

소속학부(과)	디지털콘탠츠공학과	팀명	Tense
개설 연도 및 학기	2022학년도 □1학기 □2학기	교과목명	캡스톤디자인
과제명	미니게임 제작		
과제유형	□기업연계형 캡스톤디자인	□기술이전형 캡스톤디자인	
희망금액		(기술이전금액)천원	

참여기업현황					
기업	기업명	펀웨이브	소재지	전주	
	사업자번호		주요생산품목		
담당자	성명		소속부서		
	H.P		E-mail		

참여 학생 현황

구분	이름	학부(과)	학년	학번	H.P	E-mail
팀장		디지털콘탠츠공학과	4			
팀원1		디지털콘탠츠공학과	4			
팀원2		디지털콘탠츠공학과	4			
팀원3						
팀원4						
팀원5						
팀원6						
팀원7						

집행경비내역	비목	집행내역	금액
	재료비		천원
	인쇄비		천원
	학생여비	자세히 작성	
	학생회의비	( )천원 × ( )인 × ( )회	천원
			천원
	총액		천원

위와 같이 캡스톤디자인(종합설계) 결과보고서를 제출합니다.

첨부 : 캡스톤디자인(종합설계) 과제 상세 결과보고서[별첨 1호]

2022년 6월 15일

지원학생(팀장)

사업책임자(지도교수) (인)

참여기업 담당자 (인)

원광대학교 LINC 3.0 사업단장 귀하

캡스톤디자인(기업연계프로젝트)

미니게임 자료조사 및 계획서

10조( )

목차

1. 미니게임이란?

1-1. 미니게임의 정의 및 특징

미니게임은 쉽게 말하면 게임 안의 또 다른 게임으로, 일반적으로 즉시 이해할 수 있을 정도로 직관적이고 플레이타임이 길지 않은 게임이다. 때문에 예전부터 자주 개발되어 사람들에게 익숙한 고전적인 게임들이 자주 사용되는 편이다.

미니게임의 특성상 누구나 부담 없이 즐길 수 있기 때문에 친구 · 가족 중심의 케쥬얼한 파티 게임을 만들 수 있다.
이런 게임들을 종합미니게임(Minigame compilations)라고 부른다.

※다른 게임들과 구분

구분	미니게임	캐주얼 게임	MMORPG
평균 게임 시간	3분	30분	2시간 이상
중독성	낮음		높음
유저 구성	쉽게 즐길수 있어 10대~40대까지 다양한 유저 기반		충성도 높은 반면, 상당한 시간과 노력을 통한 능력치 습득 필요성으로, 10대층의 비중 높음
특성	killing time 고객 기반, 다양한 서비스 추가를 통한 확장성 높음		콘텐츠 기반의 높은 충성도, 캐주얼 게임에 비해 확장성 낮음
진입 장벽	비교적 시장진입장벽이 낮으나, 넷마블, NHN(한게임), 네오위즈(세이게임) 등 선두 업체들의 과점화 현상		개발 기간 및 비용부담, 전문적 기술 인력등의 필요성으로 상대적으로 높은 시장진입장벽
게임 개발 기간	3~6개월		1~2년 이상
비용 리스크	개발비용부담 낮음		대형화 추세로 개발비 부담 높아 high risk, high return

하지만 현재에는 2000년대 초반 유행했던 5분이하의 짧은 플레이타임을 가진 미니게임류가 사장되었다.
대신 멀티플레이를 통해 주변 사람들과 상호작용하는 30분 이상의 보드게임 기반 미니게임 플레이 방식이 가장 대중적이다.

1-2. 미니게임의 종류

어떠한 종류의 게임이든 미니게임이 될 수 있지만 미니게임의 특성상 자주 사용되는 종류가 있다.
상기했듯 플레이가 직관적인 고전게임이 대부분이며 미니게임이다보니 당연히 난이도는 독립적인 게임보다 낮다.

ex)
*야바위 (https://namu.wiki/w/%EC%95%BC%EB%B0%94%EC%9C%84)

*탄막슈팅게임 (https://namu.wiki/w/%ED%83%84%EB%A7%89%20%EA%B2%8C%EC%9E%84?from=%ED%83%84%EB%A7%89%20%EC%8A%88%ED%8C%85%20%EA%B2%8C%EC%9E%84)

*퍼즐게임 (https://namu.wiki/w/%ED%8D%BC%EC%A6%90%20%EA%B2%8C%EC%9E%84)

*벽돌깨기 (https://namu.wiki/w/%EB%B2%BD%EB%8F%8C%EA%B9%A8%EA%B8%B0, Arcade Game: Monkey Magic (1979 Nintendo) - YouTube)

*포인트앤클릭 (https://namu.wiki/w/%ED%8F%AC%EC%9D%B8%ED%8A%B8%20%EC%95%A4%20%ED%81%B4%EB%A6%AD%20%EC%96%B4%EB%93%9C%EB%B2%A4%EC%B2%98%20%EA%B2%8C%EC%9E%84)

1-3. 미니게임 현황

1-3-1.과거

제목	액션 퍼즐 패밀리	미니게임 천국	마리오 파티
제작사	컴투스(2008)	컴투스(2005)	닌텐도(1998~)
이미지			

미니게임 천국은 2005년 첫 출시를 시작으로 지난 2010년까지 총 5개의 시리즈가 출시
조작이 간단한 원 버튼 플레이와 함께 반복의 재미가 있는 게임성으로 인기를 누림.
다운로드 1000만 건을 돌파

1-3-2.현재

제목	파티 패닉	폼멜 파티	잇테이크 투
제작사	Everglow Interactive Inc.	Rebuilt Games	Hazelight
이미지			

과거 피쳐폰 시절 접할 수 있었던 '액션 퍼즐 패밀리', '미니게임 천국'은 Killing Time용, 싱글 플레이를 하여 점수를 기록하고 다른 사람들과 경쟁하는 시스템 이였다.
하지만 스마트폰이 등장하면서 종합미니게임 장르가 쇠퇴했다.

현재에는 닌텐도에서 출시하는 마리오 파티는 기본적으로 멀티 플레이를 권장한다.
기본 베이스는 보드게임이다. 주사위를 던질때마다 미니게임을 통해 플레이어들 간에 순위를 가려 차등 보상을 받는다. 보상을 활용해 마지막까지 살아남는 플레이어가 우승한다.
마리오 파티가 흥행하고 현재 그와 같은 방식을 채택한 게임(파티 패닉, 폼멜 파티)가 흥행하고 있다.

파티 패닉의 경우 최소 34억원~최대 85억 원가량 매출 예상
폼멜 파티의 경우 최소 310억원~ 최대 775억원 가량 매출 예상

2. 미니게임 개발

2-1. 개발 배경

요즘 출시되는 대부분의 게임들은 어느 정도의 시간과 돈을 투자해야 하고, 난이도 또한 높다.
이러한 점들을 바탕으로 바쁘게 살아가는 현대인들에게 잠깐의 자투리 시간에 즐길 수 있는 미니게임 개발을 생각하게 되었다.
또한 간단하고도 재미있는 요소의 테마들을 바탕으로 게임을 개발함으로써 요즘 출시되는 게임의 복잡성, 난이도의 부담도 한층 덜어줄 수 있다.

대부분 흥행을 이끈 미니게임들은 고전게임들이다.
이번에 미니게임 개발을 통해 고전게임의 향수를 불러 일으켜 보겠습니다.

2-2. 개요

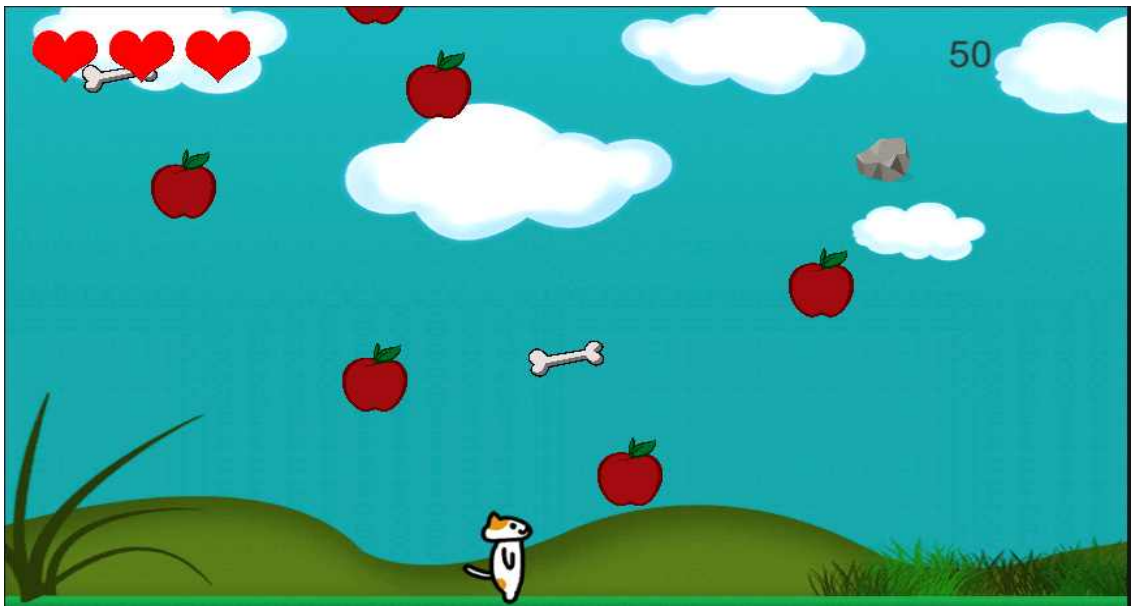
미니게임 모음형식으로 장르가 하나인 게임만 있는 것이 아닌 미니게임 천국 시리즈와 같이 쉽고 빠르게 한 번에 많은 종류의 게임들을 담는 것을 목표로 한다.

위의 게임과 같이 한게임 내에서 피하기 게임, 낚시 게임, 퍼즐 게임과 같이 조작은 단순하고 피로도가 낮은 게임들을 모아 짧은 휴식시간에 잠깐 가볍게 즐길 수 있는 것을 목표로 한다.

2-3. 테마

(1)피하기 게임

특정 오브젝트를 상대로 장애물을 피하거나 파괴하는 형식으로 진행되는 게임이다.

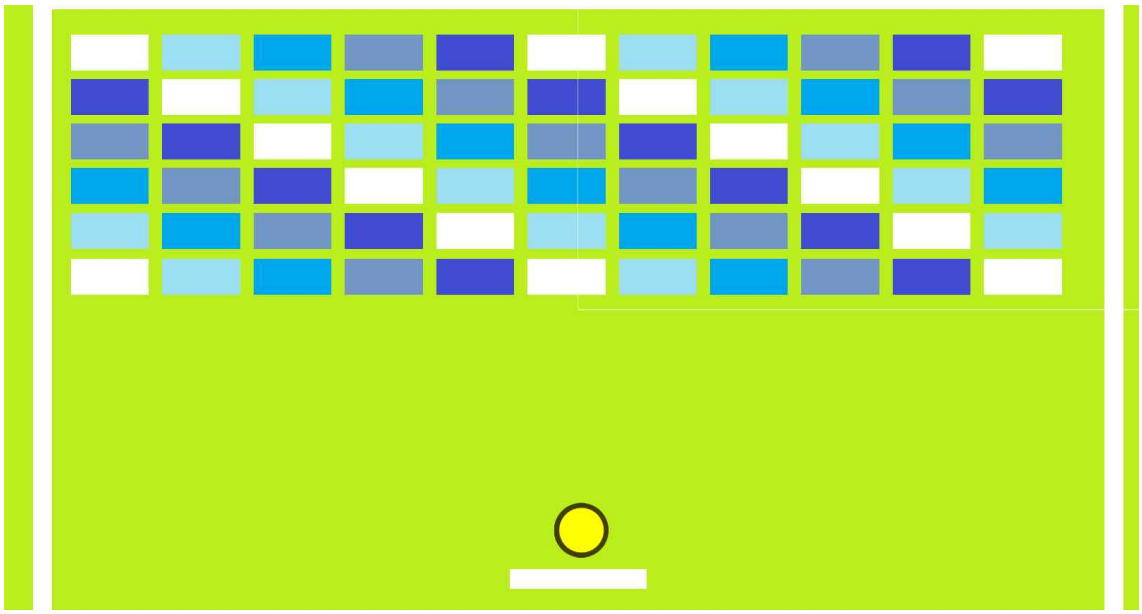


(2)점프 게임

점프를 하면서 목표지점까지 올라가는 형식으로 진행되는 게임이다.

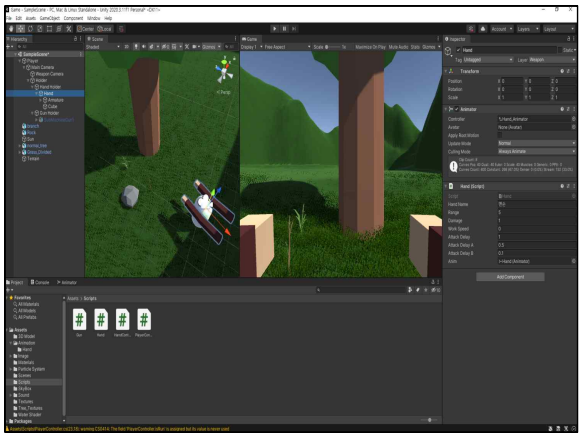
(3)벽돌 깨기 게임

특정 오브잭트를 통해 벽돌 오브젝트들을 전부 부수거나 공이 아래로 떨어지면 게임이 종료된다.



2-4. 사용 프로그램

-유니티 프로그램



2-5. 개발과정

(1)피하기 게임
1. Player 좌우 움직임
2. Enemy오브젝트 랜덤 위치 생성, 생성 후 낙하
3. Enemy오브젝트가 Player 혹은 Wall에 충돌 시 파괴
4. Player 움직임을 제한하는 벽
5. hp 및, hp UI추가
6. 게임 오버 패널 추가
7. 오른쪽 상단에 시간의 흐름에 따라 상승하는 점수 추가, 다른 속도로 떨어지는 3종류의 Enemy 추가
8. Player, Enemy 스프라이트 변경

※코드 소개※

● Enemy 랜덤 생성 코드(DropDownEnemyGen)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DropDownEnemyGen : MonoBehaviour
{
    public GameObject EnemyPrefab1;
    public GameObject EnemyPrefab2;
    public GameObject EnemyPrefab3;
    float span = 0.8f; //0.8초마다 Enemy가 떨어짐
    float delta = 0;



    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        this.delta += Time.deltaTime;
        if(this.delta>this.span)
        {
            this.delta = 0;
            //Enemy종류 1,2,3
            GameObject go1 = Instantiate(EnemyPrefab1) as GameObject;
            GameObject go2 = Instantiate(EnemyPrefab2) as GameObject;
            GameObject go3 = Instantiate(EnemyPrefab3) as GameObject;

            //-8~8값 랜덤 지정
            int px1 = Random.Range(-8, 8);
            int px2 = Random.Range(-8, 8);
            int px3 = Random.Range(-8, 8);

            //Line 34, 35, 36에서 결정한 x값, 높이 6에서 오브젝트 생성
            go1.transform.position = new Vector3(px1, 6, 0);
            go2.transform.position = new Vector3(px2, 6, 0);
            go3.transform.position = new Vector3(px3, 6, 0);
        }
    }
}
```

● Player 코드(DropDownPlayer)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DropDownPlayer : MonoBehaviour
{
    public int move_method;
    public float speed;
    public Vector2 speed_vec;

    // Start is called before the first frame update
    void Start()
    {

    }

    private void OnTriggerEnter2D(Collider2D collision)
    {

        if(collision.CompareTag("Enemy"))
        {
            HPManager.hp -= 1;
            //Debug.Log("적에 닿음"+HPManager.hp);
        }
    }

    // Update is called once per frame
    void Update()
    {
        if(HPManager.hp == 0)
        {
            Destroy(gameObject);
        }

        if (move_method == 0)
        {
            speed_vec = Vector2.zero;

            if (Input.GetKey(KeyCode.RightArrow))
            {
                speed_vec.x += speed;
            }
            if (Input.GetKey(KeyCode.LeftArrow))
            {
                speed_vec.x -= speed;
            }
            /*
            if (Input.GetKey(KeyCode.UpArrow))
            {
                speed_vec.y += speed;
            }
            if (Input.GetKey(KeyCode.DownArrow))
            {
                speed_vec.y -= speed;
            }
            */
            //transform.Translate(speed_vec);
            GetComponent<Rigidbody2D>().velocity = speed_vec;

        }

        else if (move_method == 1)
        {
            speed_vec.x = Input.GetAxis("Horizontal") * speed;
            //speed_vec.y = Input.GetAxis("Vertical") * speed;

            //transform.Translate(speed_vec);
            GetComponent<Rigidbody2D>().velocity = speed_vec;
        }
    }
}
```

● Enemy 코드(DropDownEnemy)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DropDownEnemy : MonoBehaviour
{
    public int drop_method;

    // Start is called before the first frame update
    void Start()
    {

    }

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Player"))
//Player 충돌 시 오브젝트 삭제
        {
            Dead();
        }
        if (collision.CompareTag("Wall"))
//Wall 충돌 시 오브젝트 삭제
        {
            Dead();
        }
    }

    // Update is called once per frame
    void Update()
    {
        //transform.Translate(0, -0.02f, 0); //0.1f의 속도만큼 낙하

        //y 포지션이 -5.0미만일 시 오브젝트 삭제
        if(transform.position.y < -5.0f)
        {
            Destroy(gameObject);
        }

        //drop_method == 1이면 0.01f의 속도로 아래로 떨어짐
        if (drop_method == 1)
        {
            transform.Translate(0, -0.01f, 0);
        }

        //drop_method == 2이면 0.03f의 속도로 아래로 떨어짐
        if (drop_method == 2)
        {
            transform.Translate(0, -0.03f, 0);
        }

        //drop_method == 3이면 0.05f의 속도로 아래로 떨어짐
        if (drop_method == 3)
        {
            transform.Translate(0, -0.05f, 0);
        }
    }

    //오브젝트 파괴 함수
    public void Dead()
    {
        Destroy(gameObject);
    }
}
```

● HP 코드(HPManger)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;


public class HPManager : MonoBehaviour {
    public static int hp = 3;

    //오브젝트 받아오기
    public GameObject life1;
    public GameObject life2;
    public GameObject life3;

    public Panel_GameOver panel_GameOver;

    // Use this for initialization
    void Start () {
        life1.SetActive(true);  //life1오브젝트 활성화
        life2.SetActive(true);
        life3.SetActive(true);
    }

    // Update is called once per frame
    void Update () {
        //DropDownPlayer 28번 라인에서 값을 받아옴
       switch(hp)
        {
            case 2:
                life3.SetActive(false);
//  hp가 2가되면, life1오브젝트 비활성화
                break;
            case 1:
                life2.SetActive(false);
                break;
            case 0:
                life1.SetActive(false);
                //game over
                break;
        }

        if(hp == 0)
        {

            GameOver();

        }
    }

    public void Stop()
    {
        StopAllCoroutines();
    }


    public void GameOver()
    {
        //Debug.Log("GameOver");
        Stop(); // 플레이어 스탑
        panel_GameOver.Show(); // 게임오버 팝업 띄우기
        Time.timeScale = 0.0F; //게임 정지
    }
}
```

● 게임 오버 패널 코드(Panel_GameOver)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Panel_GameOver : MonoBehaviour
{
    //public Text Text_GameResult; // 게임의 결과를 표시해줄 Text Ui
    private void Awake()
    {
        transform.gameObject.SetActive(false);
// 게임이 시작되면 GameOver 팝업 창을 보이지 않도록 한다.
    }
    public void Show() {

// int score = FindObjectOfType<ScoreText>().GetScore(); // ScoreTe
xt로 부터 현재 기록된 점수를 불러온다.
        transform.gameObject.SetActive(true);
// GameOver 팝업 창을 화면에 표시 시키고

//Text_GameResult.text = "GameSet\nScore : " + score.ToString(); //
팝업의 점수 창에 현재 점수를 표시한다.


// \n 이라는 문자는! '줄바꿈' 즉! GameSet이라는 글자 다음에 한줄 띄어라



// 라는 뜻이다.
    }




    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

● Time 코드(Timer)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Timer : MonoBehaviour
{

    public Text timeText;
    private float time;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        time += Time.deltaTime;
        timeText.text = ((int)time*10).ToString (); //초당 10점
    }
}
```

(2)점프 게임

진행상황

1. player 움직이기
2. 도착지점 깃발 설정
3. player 도착지점에 닿을 시 게임 리셋
4. CLEAR화면에서 TAB키를 누를 시 개임 다시 시작
5. 캐릭터 화면 밖으로 넘어가는 걸 방지 하기 위한 투명한 벽 생성
6. 점프 횟수 초과 시 게임 오버화면 이동 및 ui구현
7. 배경화면 추가, 일시 정지 버튼 추가, 메인화면 추가, 게임 선택 창 추가
8. 백이미지 움직이기, 백그라운드 음악 넣기
9. tab누를시 메인메뉴 이동 추가

※코드 소개※

● 카메라 코드(CameraController)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CameraController : MonoBehaviour
{
    Transform catTransform;//고양이 위치
    // Start is called before the first frame update
    void Start()
    {
        catTransform = GameObject.Find("cat").transform;//고양이 위치 할당
    }

    // Update is called once per frame
    void Update()
    {
        Vector3 catPos = catTransform.position;
        transform.position = new Vector3(transform.position.x,
            catPos.y, transform.position.z);//y축 위치를  고양이의 위치와 같도록


    }
}
```

● 움직임(이동) 코드(MOve)-(1)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class MOve : MonoBehaviour
{
    Rigidbody2D rb2D;
    Animator animator;
    const float jumpForce = 700f;//점프 힘
    const float walkForce = 30f, maxWalkSpeed = 2f;//걷는 힘
    public bool isGrounded = false;
    public int jumpCount;//점프 가능 횟수 변수

    bool jump;
    public Text jumpcount; //점프 남은 횟수텍스쳐
    bool isJumping = false;//점프상태 변수
    // Start is called before the first frame update
    void Start()
    {        Application.targetFrameRate = 120;
        rb2D = GetComponent<Rigidbody2D>();
        animator = GetComponent<Animator>();
        jumpCount = 11;
        jumpcount.text = "점프횟수"+"10";//현제 남은 점프 횟수
    }
    private void OnCollisionEnter2D(Collision2D collision)
    {
        //if (gameObject.tag == "cloud")

        //{

        //    isGrounded = true;    //Ground에 닿으면 isGround는 true

        //    jumpCount = 100;          //Ground에 닿으면 점프횟수가 2로 초기화됨

        //}
    }

    // Update is called once per frame
    void Update()
    {


        if(isGrounded)//구름에 있을 시
        {
            if(jumpCount >0)
            {
                if (Input.GetKeyDown(KeyCode.Space))//스페이스바 클릭시 점프 동작 수행
                {
                    if (isJumping == false)
                    {

                        isJumping = true;

                        GetComponent<Rigidbody2D>().AddForce(Vector3.up * 100f);


                    }
                    rb2D.AddForce(transform.up * jumpForce);
                    animator.SetTrigger("Jump");
                    jumpCount--; // 점프 카운트 감소
                    jumpcount.text = "점프횟수" + jumpCount.ToString();//점프 텍스트 표시
                }

            }
        }
```

● 움직임(이동) 코드(MOve)-(2)

```
68
69
70
71          int key = 0;//키보드 입력상태가 0
72          if (Input.GetKey(KeyCode.D)) //D입력시 오른쪽 이동
73          {
74              key = 1;
75              animator.SetTrigger("Walk");
76          }
77          else if (Input.GetKey(KeyCode.A))//A입력시 왼쪽 이동
78          {
79              key = -1;
80              animator.SetTrigger("Walk");
81          }
82
83          float speedX = Mathf.Abs(rb2D.velocity.x);
84
85          if (speedX < maxWalkSpeed)//움지임지속
86          {
87              rb2D.AddForce(transform.right * key * walkForce);
88
89          }
90
91          if (key != 0)
92          {
93              transform.localScale = new Vector3(key, 1f, 1f);
94          }
95
96          animator.speed = speedX / 2f;
97
98          if(jumpCount==0) //점프 횟수 0이 될 시 게임오버
99          {
100             SceneManager.LoadScene(4/*Game Over*/);
101         }
102
103     }
104     //
105     private void OnTriggerEnter2D(Collider2D collision)
106     {
107         //깃발에 도착시
108         SceneManager.LoadScene(6/*"Clear"*/);
109         Debug.Log("충돌감지");
110         if (collision.gameObject.tag == "Enemy")//장애물 충돌시
111         {
112
113             SceneManager.LoadScene(4);//게임오버
114         }
115
116     }
117
118
119
120 }
121
```

● Scene 코드

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CameraController : MonoBehaviour
{
    Transform catTransform;//고양이 위치
    // Start is called before the first frame update
    void Start()
    {
        catTransform = GameObject.Find("cat").transform;//고양이 위치 할당
    }

    // Update is called once per frame
    void Update()
    {
        Vector3 catPos = catTransform.position;
        transform.position = new Vector3(transform.position.x,
            catPos.y, transform.position.z);//y축 위치를  고양이의 위치와 같도록


    }
}
```

● 일시정지, 메인메뉴 코드(Btn manager)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class BTNmanager : MonoBehaviour
{
    public GameObject menuPanel;
    bool pauseActivity = false;
    // Start is called before the first frame update
    void Start()
    {
        pauseActivity =  false;

    }

    // Update is called once per frame
    void Update()
    {

    }
    public void JUMPJUMP()
    {
        SceneManager.LoadScene(2);

    }
    public void DropDownGame()
    {
        SceneManager.LoadScene(5);

    }
    public void BlockDestory()
    {
        SceneManager.LoadScene(8);

    }

    public void Pause()
    {
        if(pauseActivity==false)
        {

             Time.timeScale = 0;//시간 일시 정지
            pauseActivity = true;
            menuPanel.SetActive(true);//메뉴창 활성화
        }
        }
        else
        {
            Time.timeScale = 1;//시간정상 작동
            pauseActivity = false;//메뉴창 비활성화
            return;

        }

    }
    public void Continue()
    {
        Time.timeScale = 1;
        menuPanel.SetActive(false);//시간정상 작동
    }

    public void GameExit()
    {
        SceneManager.LoadScene("main");

    }
}

```

● 스타트버튼 코드(StrBtn)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class StrBtn : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }

    public void OnclickStart()
    {
        SceneManager.LoadScene(1);
    }
}

```

● 좌우 반복이동 코드(Reapeat)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Reapeat : MonoBehaviour
{
    Vector3 pos; ////현재위치

   public float delta = 17.0f; // 좌우 이동가능한 (x)최대값

    float speed = 0.5f; // 이동속도
    // Start is called before the first frame update
    void Start()
    {
        pos = transform.position;
    }

    // Update is called once per frame
    void Update()
    {
        Vector3 v = pos;

    v.x += delta * Mathf.Sin(Time.time * speed);

    //x축 이동
    transform.position = v;
    }
}

```

● 상하 반복이동 코드(Reapeat1)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Reapeat1 : MonoBehaviour
{
    Vector3 pos; //현재위치

    public float delta = 17.0f; // 좌우 이동가능한 (y)최대값

    float speed = 0.5f; // 이동속도
    // Start is called before the first frame update
    void Start()
    {
        pos = transform.position;
    }

    // Update is called once per frame
    void Update()
    {
        Vector3 v = pos;

        v.y += delta * Mathf.Sin(Time.time * speed);

        //y축 이동

        transform.position = v;
    }
}

```

● tab누를시 메인메뉴 이동 코드(Panel_GameOver)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Panel_GameOver : MonoBehaviour
{
    //public Text Text_GameResult; // 게임의 결과를 표시해줄 Text UI
    private void Awake()
    {
        transform.gameObject.SetActive(false); // 게임이 시작되면 GameOver 팝업 창을 보이지 않도록 한다.
    }
    public void Show() {
        // int score = FindObjectOfType<ScoreText>().GetScore(); // ScoreText로 부터 현재 기록된 점수를 불러온다.
        transform.gameObject.SetActive(true); // GameOver 팝업 창을 화면에 표시 시키고
        //Text_GameResult.text = "GameSet\nScore : " + score.ToString(); // 팝업의 점수 항목 현재 점수를 표시한다.
                                                                        // \n 이라는 문자는 '줄바꿈' 즉! GameSet이라는 글자 다음에 엔터 먹여라
                                                                        // 라는 뜻이다.
    }




    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKey(KeyCode.Tab))//tab키 입력시 게임 메인메뉴
        {
            SceneManager.LoadScene(0);
        }
    }
}

```

● game clear화면으로 이동 코드(Flag3)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Flag3 : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }

    private void OnTriggerEnter2D(Collider2D collision)
    {
        //Mainmenu
        SceneManager.LoadScene(3);


    }
}

```

(3)벽돌 깨기 게임

진행상황

1. 패들 동작 구현
2. 벽과 블록 오브젝트 배치후 태그 추가
3. 공이 튕기는 매터리얼 추가

※코드 소개※

● 볼 오브젝트 움직임 코드(BarMove)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class BarMove : MonoBehaviour
{

    public float speed;
    public Vector2 speed_vec;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

        speed_vec.x = Input.GetAxis("Horizontal") * speed;
        transform.Translate(speed_vec);
    }
}
```

● 볼 오브젝트 컨트롤러 코드(BallControll)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class BallControll : MonoBehaviour
{
    private bool firstClick = false;
    private Rigidbody2D rd;
    public int hp = 1;
    public Panel_GameOver panel_GameOver;
    // Start is called before the first frame update
    void Start()
    {

        rd = GetComponent<Rigidbody2D>();
        Time.timeScale=0.5f;
    }

    // Update is called once per frame
    void Update()
    {
        if(Input.GetKey(KeyCode.Space)&&!firstClick)
        {
            transform.parent=null;
            firstClick = true;
            rd.isKinematic = false;

            //공에 힘을 줌
            rd.AddForce(new Vector3(300,800,0));
        }

        if(hp==0)
        {
            GameOver();
        }
    }

    public void OnCollisionEnter2D(Collision2D collision)
    {
        //아래 벽에 충돌하면 공 파괴하고 게임 멈춤
        if(collision.gameObject.tag == "underbar")
        {
            //Destroy(gameObject);
            hp =0;
            Time.timeScale=0.0f;
        }

        //벽돌에 충돌하면 벽돌 파괴
        else if(collision.gameObject.tag =="block")
        {
            Destroy(collision.gameObject);
        }
        else if(collision.gameObject.tag =="movebar")
        {
            rd.AddForce(new Vector3(0,1,0));
        }
        else if(collision.gameObject.tag =="leftwall")
        {
            rd.AddForce(new Vector3(1,0,0));
        }
        else if(collision.gameObject.tag =="rightwall")
        {
            rd.AddForce(new Vector3(-1,0,0));
        }
    }

    public void GameOver()
    {

        StopAllCoroutines();
        panel_GameOver.Show(); // 게임오버 팝업 띄우기
        Time.timeScale = 0.0F; //게임 정지
    }
}
```

2-6. 실행방법

Unity build 실행파일을 통해 게임을 실행한다.
게임을 실행하면 게임 테마 선택창이 나와 원하는 테마의 게임을 클릭하여 플레이 한다.

3. 사업성

Unity AssetStore 및 Github에 업로드하여 수익을 창출한다.

캡스톤디자인 산학연계 교육 실적보고서

캡스톤디자인 교과목명 (교과목코드)	캡스톤디자인2 (기업연계프로젝트)			
캡스톤디자인 과제명	기본 게임 개발			
교육기간	2022년 03월 09일 ~ 2022년 06월 14일			
교육개요	게임에 필요한 기본 기능 개발교육			
기업체전문가	소속	편웨이브	성명	
교육내용	-개임 개발에 필요한 기초 기능 개발 교육			
교육운영결과	- 게임개발에 대한 기획, 시나리오작성 및 개발 과정에 대한 이해			

위와 같이 캡스톤디자인(과제명) 산학연계 교육 실적보고서를 제출합니다.

20 년 월 일

기업체전문가 : (인)

원광대학교 LINC 3.0 사업단장 귀하

캡스톤디자인 지도 실적 보고서(지도교수용)

캡스톤디자인 교과목명 (교과목코드)	캡스톤디자인2 (기업연계프로젝트)			
캡스톤디자인 과제명	기본 게임 개발			
지도학생	* 이름(학번) - 이름과 학번을 정확하게 모두 기재하여 주시기 바랍니다 * **팀별(프로젝트별)로 별도로 작성하여야, 업적평가시 개별로 실적 인정받습니다.**			
지도개요				
지도교수	소속	디지털콘텐츠공학과	성명	
세부 지도내용	기본 게임 개발에 대한 기본 지식 지도 - 게임 개발에 필요한 기획력 향상 - 게임 시나리오 작성과 변형 - 게임 개발 과정 이해			
수행기간	2022년 03월 09일 ~ 2022년 06월 14일			

위와 같이 캡스톤디자인(과제명)의 실적 보고서를 제출합니다.

2022년 06월 17일

지도교수 : (인)

원광대학교 LINC 3.0 사업단장 귀하