



Crazy Arcade

크레이지 아케이드 모작 게임
with Unity

Name 김나영
Tel 010-3902-6074
E-mail nayeong0121@gmail.com

목차

1. 게임 개요

2. 게임 화면

3. 구조도

4. 기능 구현 및 함수

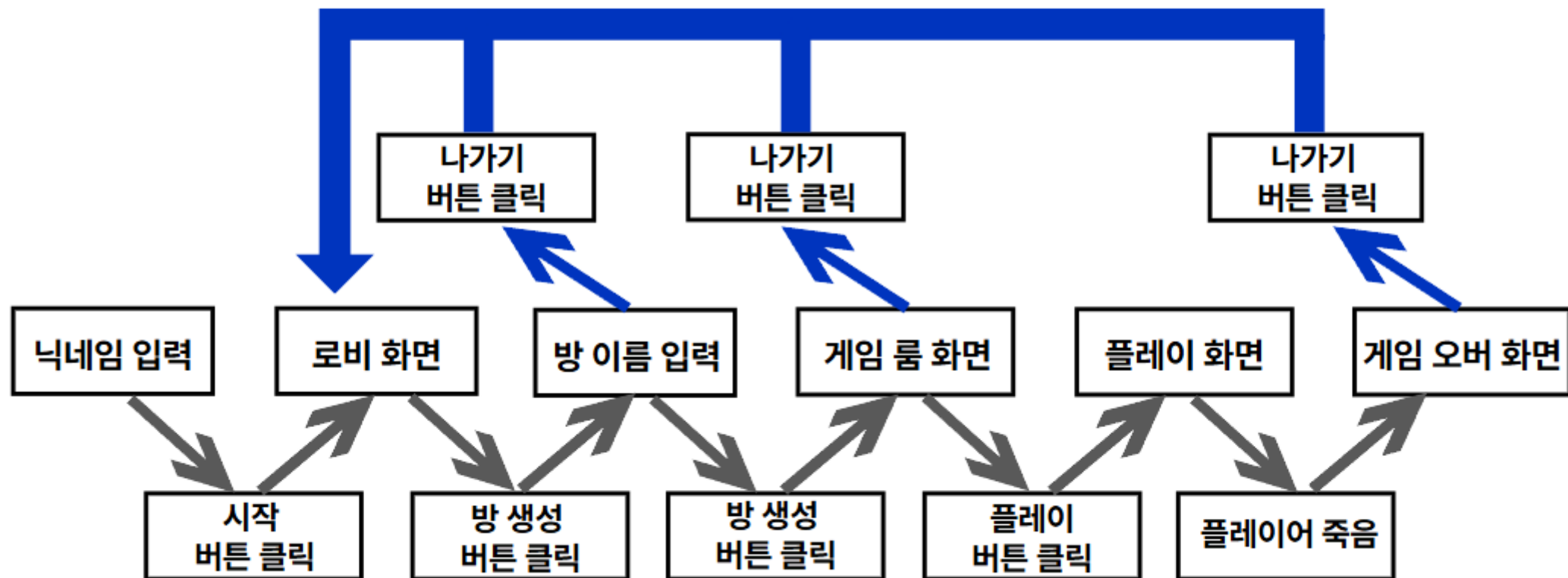
1. 게임 개요

- 언어 : C++(서버), C#(클라이언트)
- 도구 : Unity
- 플랫폼 : Windows
- [게임 영상 링크](#)
- [깃 허브 링크](#)(클라이언트)
- [깃 허브 링크](#)(IOCP 서버)
- [깃 허브 링크](#)(Node.js 서버)
- 개발 기간 : 2025.03~2025.05



- Crazy Arcade라는 넥슨 게임을 모작한 아케이드 PC 게임입니다.
- IOCP 서버를 구현하여 플레이어 2명이 물풍선을 이용하여 상대를 물방울에 가두어 터뜨리면 이기는 방식의 게임입니다.

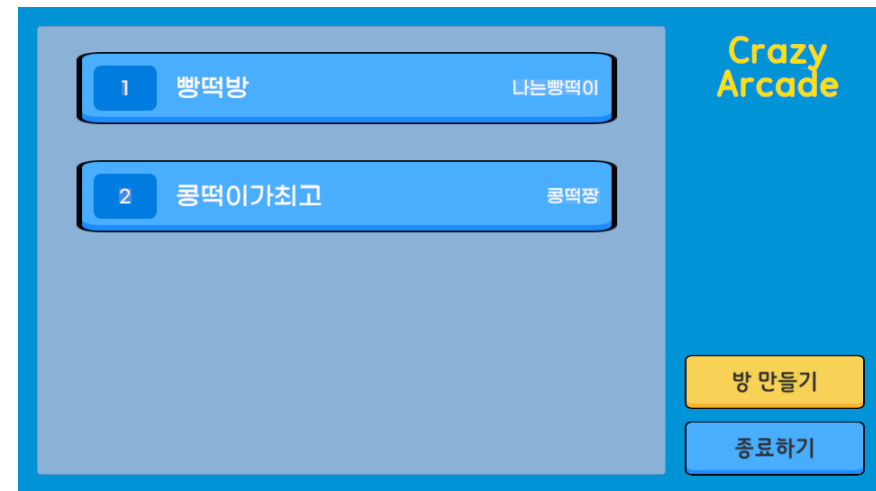
2. 구조도



3. 게임 화면



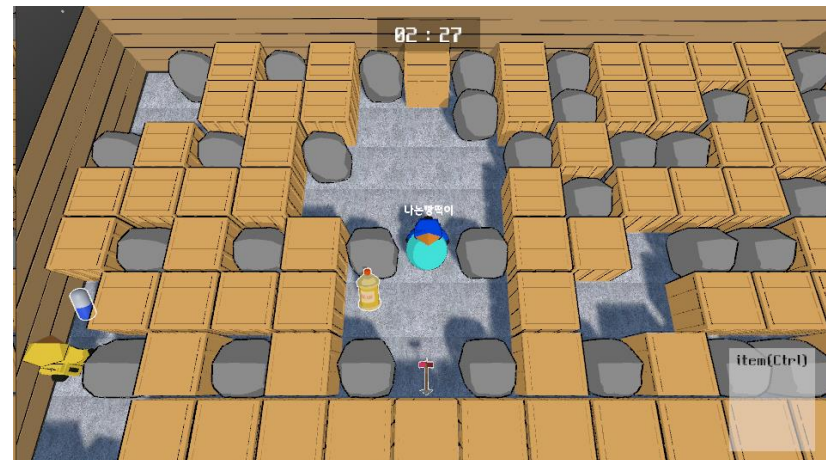
<시작 화면>



<로비 화면>



<게임 룸 화면>

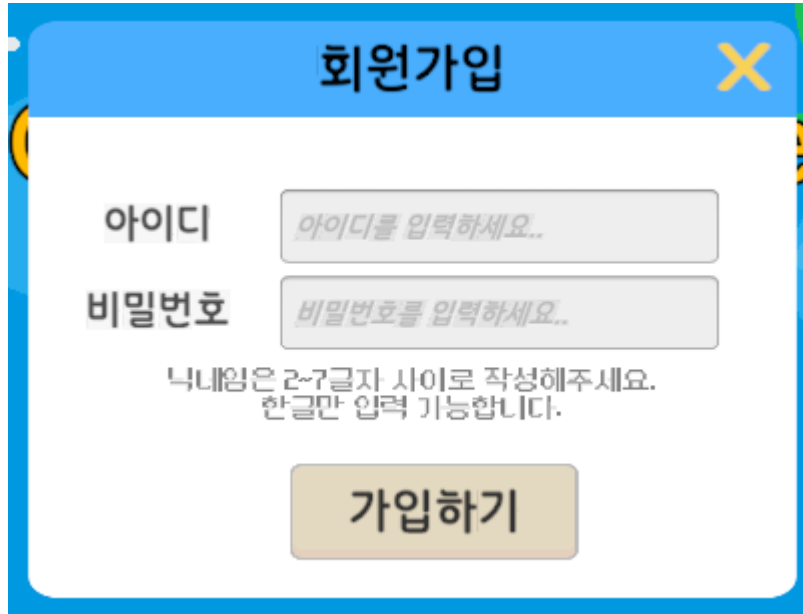


<플레이 화면>



<게임 오버 화면>

4. 기능 구현 - 회원가입 및 로그인



A screenshot of a '회원가입' (Sign Up) popup window. The window has a blue header with the title '회원가입' and a close button (X). Below the header, there are two input fields: '아이디' (ID) and '비밀번호' (Password). The ID field contains the placeholder text '아이디를 입력하세요..' and the password field contains '비밀번호를 입력하세요..'. Below these fields, there is a note: '닉네임은 2~7글자 사이로 작성해주세요. 한글만 입력 가능합니다.' (Please write the nickname between 2 and 7 characters. Only Korean characters are allowed for input). At the bottom, there is a large orange button labeled '가입하기' (Sign Up).

회원가입 팝업 창



A screenshot of the 'Crazy Arcade' login/signup screen. The title 'Crazy Arcade' is at the top in large, stylized orange letters. Below it, there are two input fields: '아이디' (ID) and '비밀번호' (Password). The ID field contains the text '어어' and the password field contains '0220'. Below these fields, there is a note: '올바른 아이디와 비밀번호를 입력하세요.' (Please enter the correct ID and password). At the bottom, there are two buttons: '로그인' (Login) and '회원가입' (Sign Up).

올바른 정보를 입력하지 않고 로그인 할 경우

- 회원가입 시 아이디와 비밀번호를 둘 다 입력하면 가입하기 버튼 활성화
- 올바르지 않은 정보 입력 시 로그인 불가 메시지 표시

4. 기능 구현 - 회원 가입 및 로그인 : 서버

```
app.post("/register", async (req, res) => {
  const { username, password } = req.body;

  if (!username || !password) {
    return res
      .status(400)
      .json({ success: false, message: "아이디 또는 비밀번호 누락" });
  }

  try {
    const hashedPassword = await bcrypt.hash(password, 10);
    const sql = "INSERT INTO users (username, password) VALUES(?, ?)";

    db.query(sql, [username, hashedPassword], (err, result) => {
      if (err) {
        if (err.code === "ER_DUP_ENTRY") {
          return res
            .status(409)
            .json({ success: false, message: "중복 아이디" });
        }
        return res.status(500).json({ success: false, message: "서버 오류" });
      }
      console.log(`username : ${username}, password : ${password}`);
      return res.json({ success: true, message: "회원가입 성공" });
    });
  } catch (error) {
    res.status(500).json({ success: false, message: "비밀번호 암호화 실패" });
  }
});
```

- Node.js로 구현한 회원가입

```
app.post("/login", (req, res) => {
  const { username, password } = req.body;

  if (!username || !password)
    return res
      .status(400)
      .json({ success: false, message: "아이디 또는 비밀번호 누락" });

  const sql = "SELECT * FROM users WHERE username = ?";
  db.query(sql, [username], async (err, results) => {
    if (err)
      return res.status(500).json({ success: false, message: "서버 오류" });
    if (results.length === 0)
      return res
        .status(401)
        .json({ success: false, message: "존재하지 않는 아이디입니다." });

    const user = results[0];
    const isMatch = await bcrypt.compare(password, user.password);

    if (!isMatch) {
      return res.status(401).json({ success: false, message: "비밀번호 틀림" });
    }
    res.json({ success: true, message: "로그인 성공", user_id: user.user_id });
  });
});
```

- Node.js로 구현한 로그인

4. 기능 구현 - 회원 가입 및 로그인 : 클라이언트

```
IEnumerator RegisterUser(string username, string password)
{
    string url = "http://localhost:3000/register";

    RegisterRequest requestData = new RegisterRequest
    {
        username = username,
        password = password
    };

    string jsonData = JsonUtility.ToJson(requestData);
    byte[] bodyRaw = System.Text.Encoding.UTF8.GetBytes(jsonData);

    UnityWebRequest request = new UnityWebRequest(url, "POST");
    request.uploadHandler = new UploadHandlerRaw(bodyRaw);
    request.downloadHandler = new DownloadHandlerBuffer();
    request.SetRequestHeader("Content-Type", "application/json");

    yield return request.SendWebRequest();

    string responseText = request.downloadHandler.text;

    if(request.result == UnityWebRequest.Result.Success)
    {
        Debug.Log("회원가입 성공 " + request.downloadHandler.text);
        failedCreateAccountText.SetActive(false);
        gameObject.SetActive(false);
    }
}
```

- 회원가입 요청

```
IEnumerator LoginToServer(string username, string password)
{
    string url = "http://localhost:3000/login";

    LoginRequest requestData = new LoginRequest
    {
        username = username,
        password = password
    };

    string json = JsonUtility.ToJson(requestData);
    byte[] bodyRaw = System.Text.Encoding.UTF8.GetBytes(json);

    UnityWebRequest request = new UnityWebRequest(url, "POST");
    request.uploadHandler = new UploadHandlerRaw(bodyRaw);
    request.downloadHandler = new DownloadHandlerBuffer();
    request.SetRequestHeader("Content-Type", "application/json");

    yield return request.SendWebRequest();

    string responseText = request.downloadHandler.text;

    if(request.result == UnityWebRequest.Result.Success)
    {
        LoginResponse res = JsonUtility.FromJson<LoginResponse>(responseText);
        if (res.success)
        {
            Debug.Log("로그인 성공");
            GameObject player = GameObject.FindGameObjectWithTag(Define.PlayerTag);
            player.GetComponent<PlayerController>().ChangeName(username);
            onStartButtonClicked?.Invoke();
        }
    }
}
```

- 로그인

4. 기능 구현 - 화면 동기화



<플레이어1 화면>



<플레이어2 화면>

- 같은 방에 있는 플레이어들의 움직임, 이름, 색상, 아이템 동기화
- 로비 화면과 게임 룸 화면 동기화

4. 기능 구현 - 화면 동기화 : 서버

```
case PacketType::PlayerInfo:
{
    std::lock_guard<std::mutex> guard(mLock);
    PlayerInfo playerInfo;
    std::memcpy(&playerInfo, packetData.pPacketData, sizeof(PlayerInfo));
    playerInfo.isValid = 1;

    gPlayers[packetData.SessionIndex] = playerInfo;

    RoomInfo* myRoom = FindRoomByPlayerIndex(packetData.SessionIndex);
    if (myRoom != nullptr) {
        int recvPlayerIndex;
        if (myRoom->createPlayerIndex == packetData.SessionIndex && myRoom->otherPlayerIndex != -1)
            recvPlayerIndex = myRoom->otherPlayerIndex;
        else if (myRoom->otherPlayerIndex == packetData.SessionIndex)
            recvPlayerIndex = myRoom->createPlayerIndex;
        SendMsg(recvPlayerIndex, sizeof(PlayerInfo), reinterpret_cast<char*>(&playerInfo));
    }

    break;
}
```

- IOCP 서버를 이용해 서버에서 플레이어들의 정보를 수신
- 서버에서 같은 방에 있는 다른 플레이어들에게 받은 정보 송신

```
case PacketType::RoomInfo:
{
    std::lock_guard<std::mutex> guard(mLock);
    RoomInfo roomInfo;
    std::memcpy(&roomInfo, packetData.pPacketData, sizeof(RoomInfo));
    roomInfo.roomIndex = mRoomNum;
    gRooms.insert(make_pair(mRoomNum++, roomInfo));
    break;
}

{
    std::lock_guard<std::mutex> guard(mLock);
    for (auto& room : gRooms){
        for (auto& otherPlayer : gPlayers) {
            SendMsg(otherPlayer.first, sizeof(RoomInfo),
                    reinterpret_cast<char*>(&room.second));
        }
    }
}
```

- 서버에서 방 정보를 모든 플레이어에게 송신

4. 함수 구현 - 화면 동기화 : 클라이언트

```
void RecvOtherInfo(PlayerInfo otherPlayerInfo)
{
    SpawnController spawnController = GameObject.Find("Spawner").GetComponent<SpawnController>();
    if (!_isOtherPlayerSpawned)
    {
        _isOtherPlayerSpawned = true;
        _otherPlayer = spawnController.SpawnOtherPlayer();

        _otherPlayer.name = $"Player{otherPlayerInfo.index}";
        _otherPlayer.GetComponent<PlayerController>().PlayerIndex = otherPlayerInfo.index;
        PlayerController playerController = _otherPlayer.GetComponent<PlayerController>();
        playerController.IsSelf = false;
        playerController.InitNameObject();
        playerController.ChangeName(otherPlayerInfo.name);
    }
    _otherPlayer.GetComponent<PlayerColor>().Color = otherPlayerInfo.color;
    if (_otherPlayer != null && otherPlayerInfo.index != _playerInfo.index)
    {
        _otherPlayer.transform.position = new Vector3(otherPlayerInfo.posX, otherPlayerInfo.posY, otherPlayerInfo.posZ);
        _otherPlayer.transform.rotation =
            Quaternion.Lerp(_otherPlayer.transform.rotation,
                Quaternion.Euler(otherPlayerInfo.rotX, otherPlayerInfo.rotY, otherPlayerInfo.rotZ), Time.deltaTime * 10);
    }
}
```

- 서버로부터 같은 방의 다른 플레이어 정보를 수신하는 함수

```
void SetRoom(RoomInfo room)
{
    if (!GameManager.instance.IsGameStart)
    {
        if (room.playerNum == 0)
        {
            UIManager.instance.ShowRoom(room);
            return;
        }
        if (_playerInfo.index == room.createPlayerIndex)
        {
            bool isCanPlay = (room.otherPlayerIndex != -1);
            UIManager.instance.SetRoom(room);
            UIManager.instance.SetPlayButton(isCanPlay);
        }
        else if (_playerInfo.index == room.otherPlayerIndex)
        {
            UIManager.instance.JoinRoom(room);
            UIManager.instance.SetPlayButton(false);
        }
        else
            UIManager.instance.ShowRoom(room);
    }
}
```

- 서버로부터 받은 방 정보에 따라 처리하는 함수

4. 기능 구현 - 아이템



물풍선 증가
아이템

물풍선 1개 증가



물줄기 증가
아이템

물줄기 1칸 증가



물줄기 최대
증가 아이템

물줄기 최대 증가

- 나무 블록 제거 시 생성되는 아이템
- 아이템 획득 시 해당 아이템의 효과 획득

4. 기능 구현 - 아이템



속도 증가
아이템

속도 증가



속도 최대
증가 아이템

속도 최대 증가



감전 아이템

일정 시간 동안
이동 불가



탈것 아이템

탈것을 타고
물줄기에 맞으면
탈것이 없어지는 대신
물풍선에 갇히지 않음

4. 기능 구현 - 아이템



다트 아이템

물풍선에 다트를 던져
물풍선 폭파 가능



글루 아이템

바닥에 거미줄을 생성해
거미줄을 밟으면
일정 시간동안 속도 저하



점프 아이템

점프를 통해
물줄기 피하기 가능



검 아이템

물풍선에 갇혔을 때
빠져나오기 가능

컨트롤 아이템을 먹으면 컨트롤을 눌러 아이템 사용 가능

4. 함수 구현 - 아이템 생성

```
void OnRecvItemType(Vector3 blockPos, int itemType)
{
    if (itemType != (int)Define.ItemType.None)
    {
        StartCoroutine(SpawnItemAfterDelay(blockPos, itemType));
    }
}
```

참조 1개

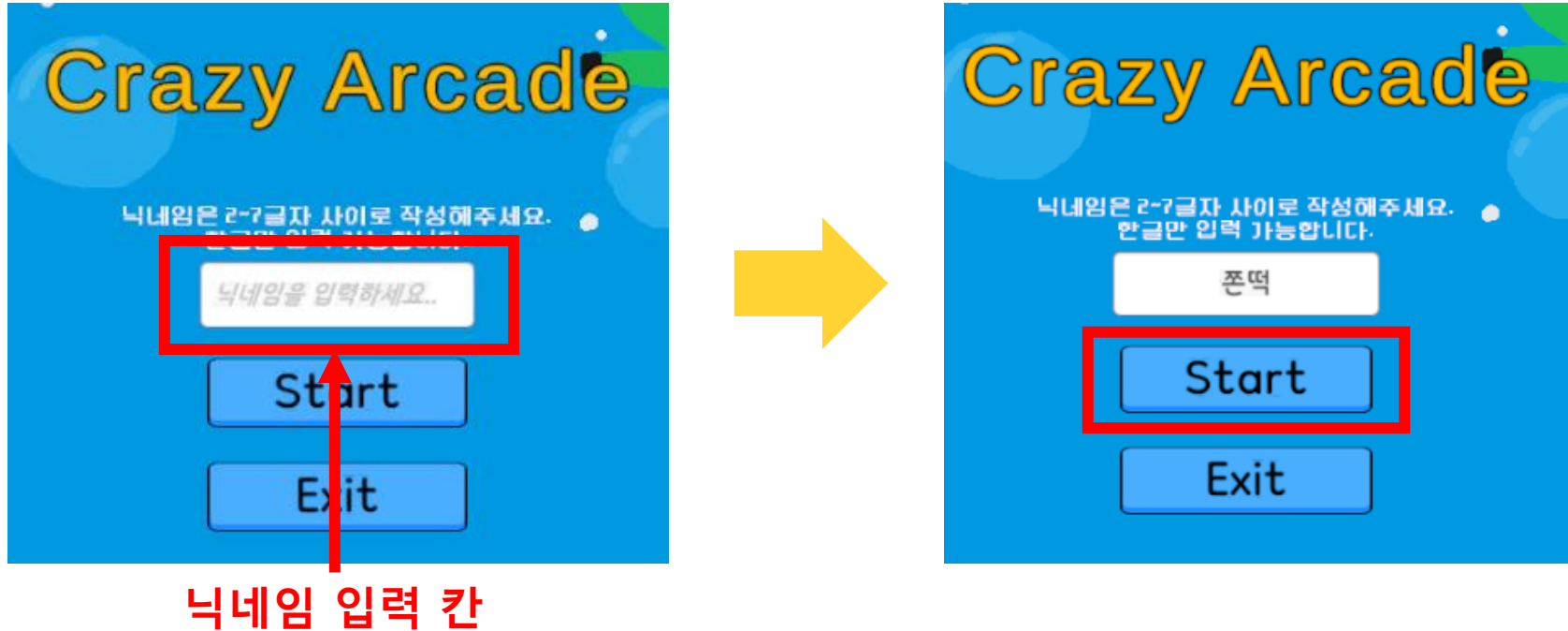
```
IEnumerator SpawnItemAfterDelay(Vector3 blockPos, int itemType)
{
    yield return _delayTime;
    GameObject item = GetItem(itemType);
    item.transform.parent = _itemPool.transform;
    item.transform.position = blockPos + new Vector3(0, 1f, 0);
    item.SetActive(true);
}
```

참조 1개

```
GameObject GetItem(int itemType)
{
    if (_itemGroup.ContainsKey(itemType))
    {
        foreach (var item in _itemGroup[itemType])
        {
            if (!item.activeSelf)
                return item;
        }
    }
    GameObject newItem = Instantiate(itemPrefabs[itemType]);
    if (!_itemGroup.ContainsKey(itemType))
    {
        List<GameObject> gameObjects = new();
        _itemGroup.Add(itemType, gameObjects);
    }
    _itemGroup[itemType].Add(newItem);
    return newItem;
}
```

서버로부터 아이템 타입을 수신하여 각 위치에 아이템을 생성하는 로직

4. 기능 구현 - 플레이어 닉네임 입력



- 메인 화면에서 자신의 닉네임 입력
- 한글만 입력 가능, 2~7글자의 닉네임 입력 시 게임 시작 가능

4. 함수 구현 - 플레이어 닉네임

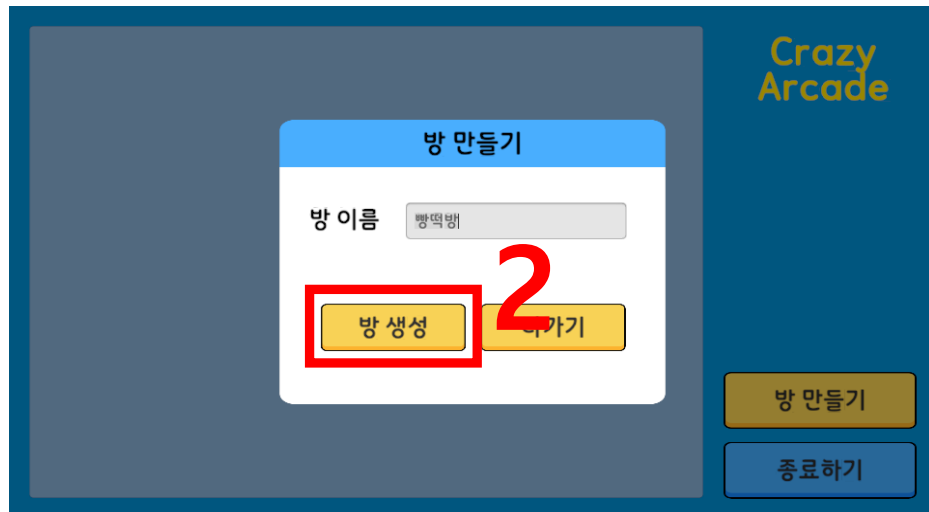
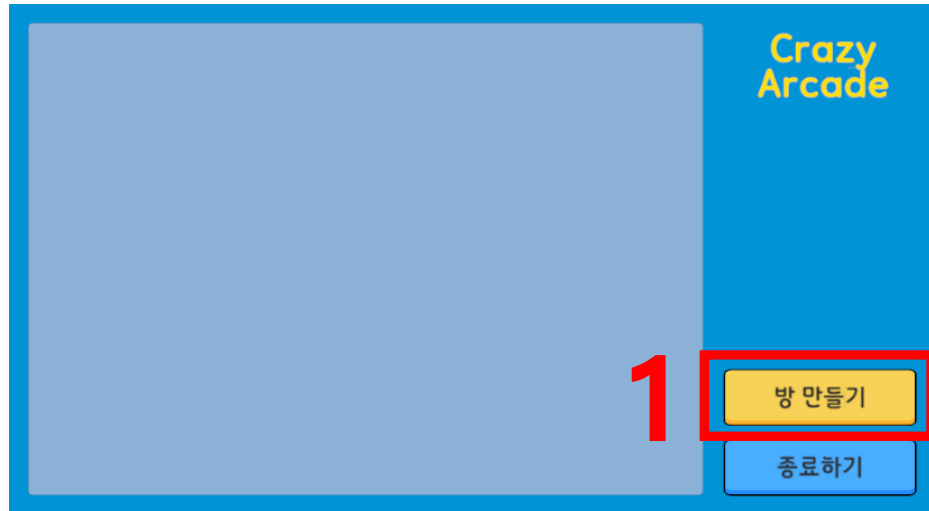
```
private void Start()
{
    StartButton.onClick.AddListener(OnStartButtonClick);
    QuitButton.onClick.AddListener(OnExitButtonClick);
    input = playerNameInput.GetComponent<TMP_InputField>();
    input.onValueChanged.AddListener((word) => input.text = Regex.Replace(word, @"[^가-힣]", ""));
}

● Unity 메시지 | 참조 0개
private void Update()
{
    playerName = input.text;
    if (playerName != null && playerName.Length >= 2 && playerName.Length <= 7){
        GameObject player = GameObject.FindGameObjectWithTag(Define.PlayerTag);
        player.GetComponent<PlayerController>().ChangeName(playerName);
        _isInputPlayerName = true;
    }
}

참조 1개
public void OnStartButtonClick()
{
    if (_isInputPlayerName)
    {
        OnStartButtonClicked?.Invoke();
    }
}
```

조건에 맞는 닉네임 입력 시
게임 시작 버튼 활성화하는 로직

4. 기능 구현 - 방 생성



- 방 생성 : 로비 화면에서 방 만들기 버튼 클릭
- 한 명의 플레이어만 방에 있을 경우 Play 버튼 비활성화

4. 함수 구현 - 방 생성

```
void OnCreateRoomButtonClick()
{
    UnityClient.instance.SendCreateRoom(input.text);
    OnCreateRoomButtonClicked?.Invoke(input.text);
}
```

- 방 이름 입력 방 생성 버튼을 클릭하면
서버에게 방 정보 전송하는 함수

```
참조 2개
public void SetRoom(RoomInfo room, bool isJoin = false)
{
    _roomNum = room.roomNum;
    roomNameText.text = room.roomName;
    roomNumText.text = (room.roomNum+1).ToString() + "번 방";
    playerNameTexts[0].text = room.createPlayerName;
    playerNameTexts[1].text = room.otherPlayerName;

    if (_uiPlayer == null) _uiPlayer = GetUIPlayer();

    _uiPlayer.transform.position = _createUIPlayerPos;

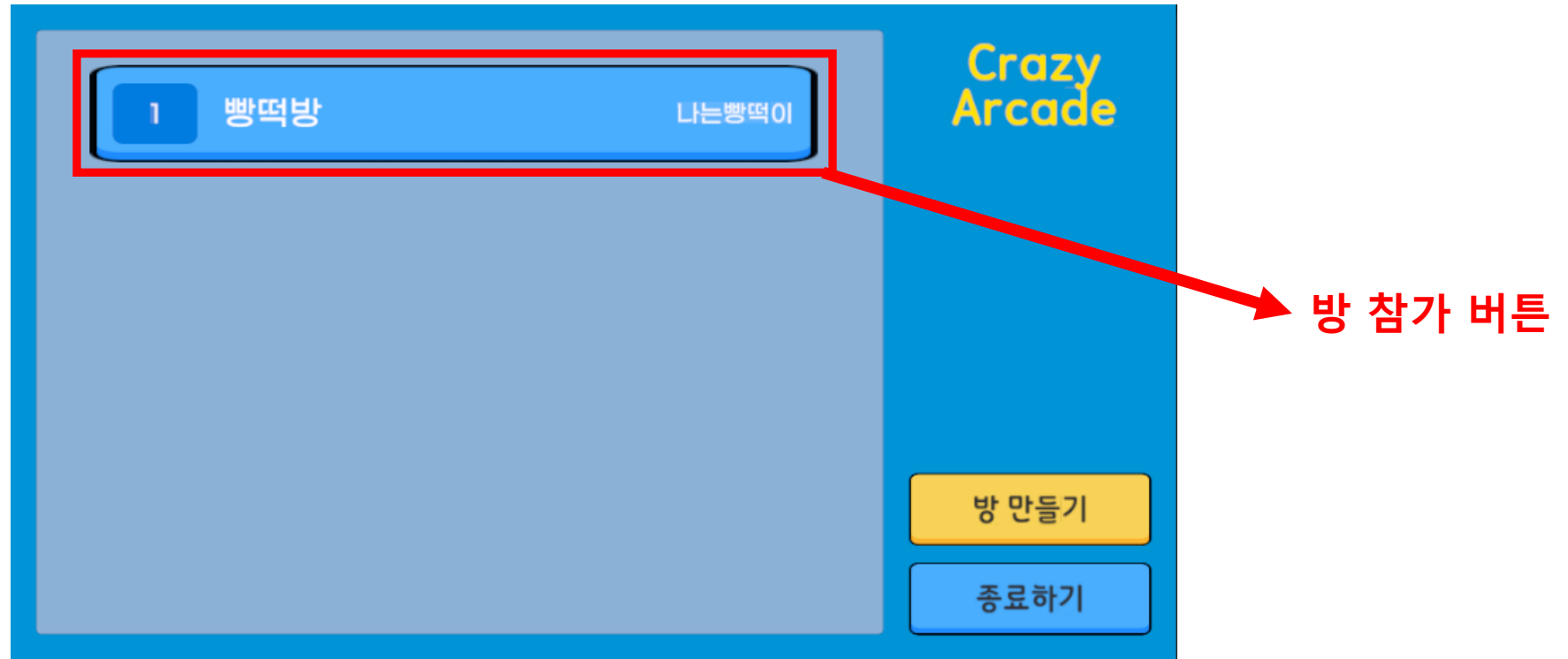
    if (room.otherPlayerIndex != -1)
    {
        if (_uiOtherPlayer == null) _uiOtherPlayer = GetUIPlayer();
        else if (!_uiOtherPlayer.activeSelf) _uiOtherPlayer.SetActive(true);

        _uiOtherPlayer.transform.position = _otherUIPlayerPos;

        if (isJoin)
        {
            _uiPlayer.transform.position = _otherUIPlayerPos;
            _uiOtherPlayer.transform.position = _createUIPlayerPos;
        }
    }
    else
    {
        if (_uiOtherPlayer != null) _uiOtherPlayer.SetActive(false);
    }
}
```

- 방 이름 & 플레이어 정보 &
서버에게 받은 방 번호를 보여주는 함수

4. 기능 구현 - 방 참가



- 다른 플레이어가 방 생성시 로비에 해당 방 이름&방을 생성한 플레이어의 이름 표시
- 방 버튼 클릭 시 해당 방에 참가
- 방에 인원이 다 찰 경우 다른 플레이어의 로비 화면에 해당 방 미표기

4. 함수 구현 - 방 표시 및 제거

```
public void CreateRoomList(int roomNum, string roomName, string createPlayerName)
{
    GameObject room = FindRoom(roomNum);
    if (room == null)
    {
        GameObject newRoom = Instantiate(roomPrefab, contentTransform);
        newRoom.GetComponent<UIRoomList>().SetRoom(roomNum, roomName, createPlayerName);
        newRoom.GetComponent<UnityEngine.UI.Button>().onClick.AddListener(
            () => OnJoinRoomButtonClicked(roomNum));
    }
    else
        room.SetActive(true);
}

//방을 안 보이게 하는 함수
참조 1개
public void HideRoomList(int roomNum)
{
    GameObject room = FindRoom(roomNum);
    if(room !=null)
        room.SetActive(false);
}

//방을 삭제하는 함수
참조 1개
public void DeleteRoomList(int roomNum)
{
    GameObject room = FindRoom(roomNum); //방 번호에 맞는 방을 찾아 지운다.
    Destroy(room);
}
```

방 인원에 따라 방 표시 여부 결정

4. 기능 구현 - 방 구현



- 새로 참가한 플레이어의 이름과 캐릭터가 화면에 표시
- 방장이 플레이 버튼을 눌러 게임 시작 가능
- 색 변경 버튼을 눌러 플레이어 색 변경 가능

4. 기능 구현 - 플레이 화면



<플레이어 화면>

- 스페이스바 : 해당 위치에 물풍선 생성
- 방향키 : 플레이어 이동
- 컨트롤키 : 갖고 있는 컨트롤 아이템 사용

4. 함수 구현 - 플레이어 이동

```
public void Move(float moveSpeed)
{
    float turnSpeed = 10;

    if (Input.GetButton(Define.Horizontal))
    {
        _dir.x = Input.GetAxisRaw(Define.Horizontal);
        _dir.z = 0;
        _movement = new Vector3(Input.GetAxisRaw(Define.Horizontal), _rigidbody.linearVelocity.y, 0);
        _lastMove = new Vector3(_dir.x, 0, _dir.z).normalized;
        _isFirstMoved = false;
    }
    else if (Input.GetButton(Define.Vertical))
    {
        _dir.x = 0;
        _dir.z = Input.GetAxisRaw(Define.Vertical);
        _movement = new Vector3(0, _rigidbody.linearVelocity.y, Input.GetAxisRaw(Define.Vertical));
        _lastMove = new Vector3(_dir.x, 0, _dir.z).normalized;
        _isFirstMoved = false;
    }
    else if (_isFirstMoved) //처음 스폰될 때
    {
        _movement = Vector3.zero;
        _dir.z = -1;
    }
    else
    {
        _movement = Vector3.zero;
    }

    _toRotation = Quaternion.LookRotation(_dir, Vector3.up);
    transform.rotation = Quaternion.Lerp(transform.rotation, _toRotation, turnSpeed * Time.fixedDeltaTime);

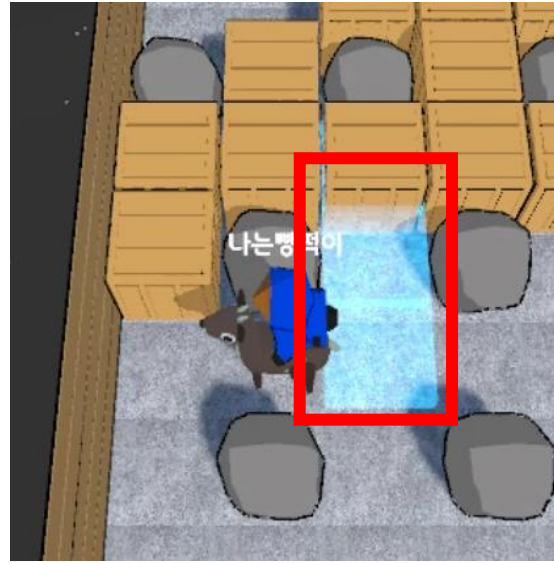
    _rigidbody.MovePosition(_rigidbody.position + _movement.normalized * moveSpeed * Time.fixedDeltaTime);
}
```

방향키에 따라 플레이어 이동

4. 기능 구현 - 물풍선과 물줄기



물풍선



물줄기



돌

- 물풍선 생성 시 3초 뒤에 터지고 물줄기 동서남북 생성
- 물줄기가 나무 블록에 닿으면 해당 나무 블록 제거(돌은 제거 불가능)

4. 함수 구현 - 물풍선과 물줄기

```
IEnumerator FireWaterBalloon()
{
    if (!_isTrapped && !_playerMove.IsJumping)
    {
        Vector3 waterBalloonPos = Util.GetRevisedPosition(transform.position);
        UnityClient.instance.SendWaterBalloonPos(waterBalloonPos);

        if(waterBalloonPos != this.waterBalloonPos.Find(w => w == waterBalloonPos))
        {
            this.waterBalloonPos.Add(waterBalloonPos);
            _waterBalloonController.SpawnWaterBalloon(waterBalloonPos);
            _waterBalloonCount--;
            yield return _waterBalloonActiveTime;
            _waterBalloonCount++;
            this.waterBalloonPos.Remove(waterBalloonPos);
        }
    }
}
```

- 스페이스바 클릭 시 물풍선을 생성하는 함수

```
IEnumerator WaterballoonBomb()
{
    yield return waterballoonBomb;
    transform.position = new Vector3(transform.position.x, 1, transform.position.z);
    OnWaterBalloonBurst?.Invoke(transform, _balloonLength);
    gameObject.SetActive(false);
}
```

- 물풍선 생성 3초 후
물줄기 생성 이벤트 호출하는 함수

4. 함수 구현 - 물풍선과 물줄기

```
void SpawnWaterCurrents(Transform waterBalloonPos, int waterCurrentLegnth)
{
    //중앙 물줄기

    foreach(var dir in directions)
    {
        RaycastHit hit;
        bool isHit = Physics.Raycast(waterBalloonPos.position, dir, out hit,
            waterCurrentLegnth * 2-1, LayerMask.GetMask("Block"));

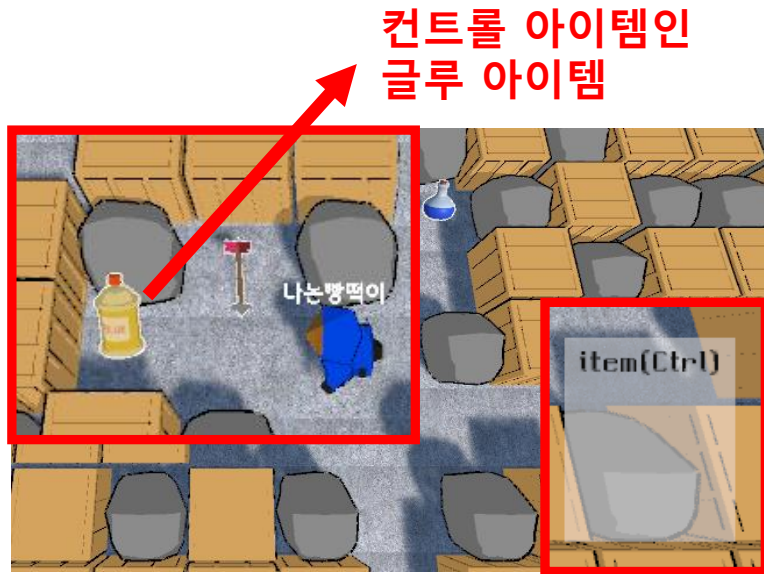
        int maxCurrent = isHit ? ((int)Mathf.Round(hit.distance) + 1) / 2 : waterCurrentLegnth;

        maxCurrent = isHit&&hit.collider.CompareTag(Define.UnbreakableBlockTag) ? maxCurrent - 1 : maxCurrent;
        for (int i = 1; i <= maxCurrent; i++)
        {
            GameObject waterCurrent = GetWaterCurrent();
            waterCurrent.transform.position = waterBalloonPos.position + dir * i * 2;
            waterCurrent.SetActive(true);
        }
    }

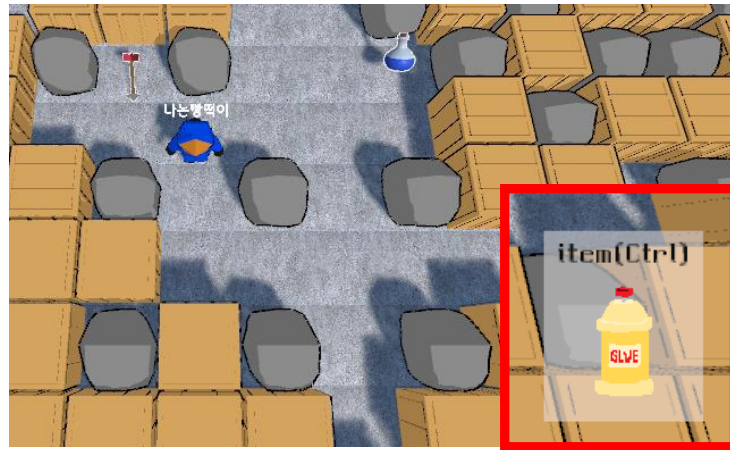
    GameObject waterCurrentsCenter = GetWaterCurrent();
    waterCurrentsCenter.transform.position = waterBalloonPos.position;
    waterCurrentsCenter.SetActive(true);
}
```

플레이어의 물줄기 길이만큼
물줄기 생성하는 함수

4. 기능 구현 - 물풍선과 물줄기



<컨트롤 아이템 획득 전>



<컨트롤 아이템 획득 후>



<컨트롤 아이템 사용 후>

- 컨트롤 아이템 획득 시 화면 오른쪽 아래에 해당 컨트롤 아이템 이미지 표시
- 컨트롤 키 클릭 시 각 컨트롤 아이템 기능에 맞는 효과 생성

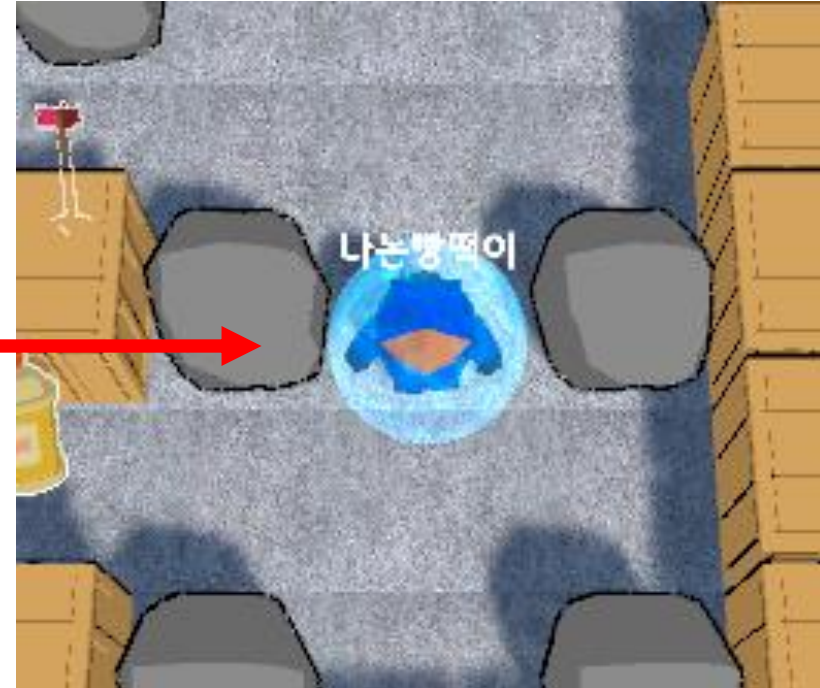
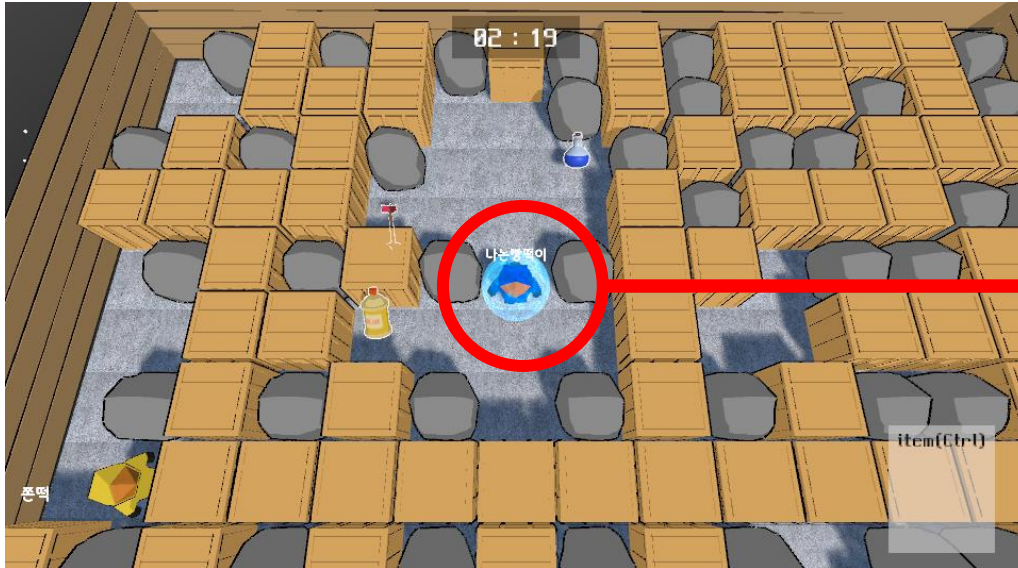
4. 함수 구현 - 컨트롤 아이템 사용

```
void UseCtrlItem()
{
    bool isUseCtrlItem = false;
    switch (_ctrlItem)
    {
        case Define.CtrlItemType.JumpItem:
            if (!_isTrapped && !_playerMove.IsGotVehicle)
            {
                _playerMove.IsJumping = true;
                StartCoroutine(_playerMove.Jump());
                isUseCtrlItem = true;
            }
            break;
        case Define.CtrlItemType.GlueItem:
            if (!_isTrapped)
            {
                UseGlueItem();
                isUseCtrlItem = true;
            }
            break;
        case Define.CtrlItemType.DartItem:
            if (!_isTrapped)
            {
                UseDartItem();
                isUseCtrlItem = true;
            }
            break;
        case Define.CtrlItemType.SwordItem:
            if (_isTrapped && !_playerMove.IsGotVehicle)
            {
                UnityClient.instance.SendUseSword();
                UseSwordItem();
                isUseCtrlItem = true;
            }
            break;
    }
    if (isUseCtrlItem)
    {
        _ctrlItem = Define.CtrlItemType.None;
        UIManager.instance.UIPlay.UpdateCtrlItemImage(Define.CtrlItemType.None);
    }
}
```

컨트롤 키 클릭 시 사용할 컨트롤 아이템 있으면
호출되는 함수

2. 기능 구현 - 플레이 화면

물풍선에 갇힌
플레이어



- 플레이어가 물줄기에 닿으면 플레이어는 물풍선에 갇힘
- 검 아이템이 없으면 플레이어는 4초 후에 자동으로 죽음
- 플레이어가 물풍선에 갇혔을 때 다른 플레이어와 부딪히면 바로 죽음

4. 함수 구현 - 플레이어 죽음

```
IEnumerator PlayerDeadTime()
{
    yield return _playerTrappedTime;
    if (gameObject.GetComponentInParent<PlayerController>().IsSelf)
    {
        GameManager.instance.GameOver(Define.WinType.Lose);
        gameObject.SetActive(false);
    }
    else
    {
        GameManager.instance.GameOver(Define.WinType.Win);
        gameObject.SetActive(false);
    }
}
```

- 플레이어가 물풍선에 갇히면 실행되는 함수

```
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag(Define.PlayerTag))
    {
        if (gameObject.transform.parent == other.transform) return;
        if (other.transform.GetComponentInParent<PlayerController>().IsSelf)
        {
            GameManager.instance.GameOver(Define.WinType.Win);
            gameObject.SetActive(false);
        }
        else
        {
            GameManager.instance.GameOver(Define.WinType.Lose);
            gameObject.SetActive(false);
        }
    }
}
```

- 플레이어가 물풍선에 갇혔을 때
다른 플레이어와 부딪히면 실행되는 함수

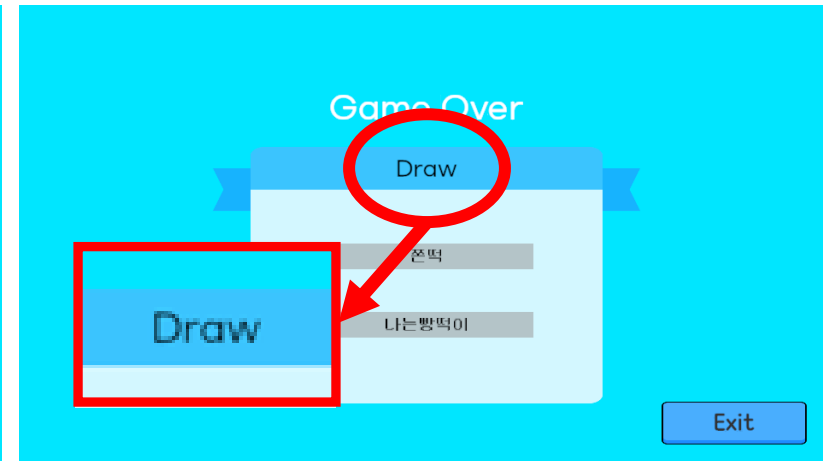
2. 기능 구현 - 게임 오버 화면



<이겼을 때 화면>



<졌을 때 화면>



<3분이 지나 비겼을 경우의 화면>

- 승리 여부에 따라 **Win!**이나 **Lose** 단어가 뜨게 됩니다.
- 모든 플레이어가 죽지 않고 3분이 지나면 **Draw**라는 단어가 뜨게 됩니다.
- 게임 오버 화면이 뜨고 5초 후에 게임 룸으로 다시 돌아가게 됩니다.

4. 함수 구현 - 게임 오버 화면

```
public void IsPlayerWin(Define.WinType winType)
{
    UIPlayGameObject.SetActive(false);
    switch (winType)
    {
        case Define.WinType.Win:
            UIGameOver.isWinText.text = "Win!";
            break;
        case Define.WinType.Lose:
            UIGameOver.isWinText.text = "Lose";
            break;
        case Define.WinType.Draw:
            UIGameOver.isWinText.text = "Draw";
            break;
    }
    UIGameOverGameObject.SetActive(true);
}
```

- 플레이어 승리 여부에 따라
게임 오버 화면에 표시될 텍스트 지정하는 함수

```
void DisplayPlayer()
{
    GameObject[] players = GameObject.FindGameObjectsWithTag(Define.PlayerTag);
    for(int i=0; i< players.Length; i++)
    {
        string playerName = players[i].GetComponent<PlayerController>().Name;
        playerNames[i].text = playerName;
    }
    StartCoroutine(ActiveGameOver());

    참조 1개
    IEnumerator ActiveGameOver()
    {
        yield return _activeGameOverTime;
        UIManager.instance.BackToGameRoom();
    }
}
```

- 방에 있는 플레이어 이름들을 보여주고
5초 후에 게임 룸 화면으로 돌아가는 로직

감사합니다.