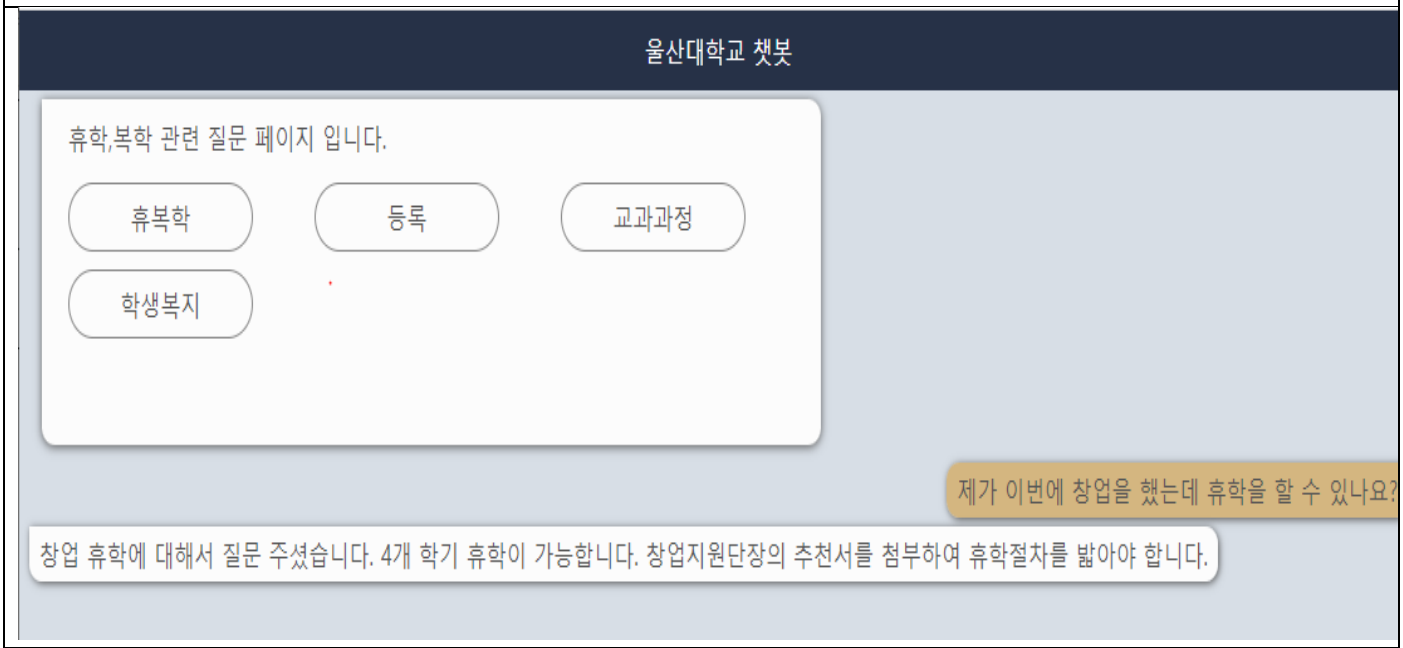
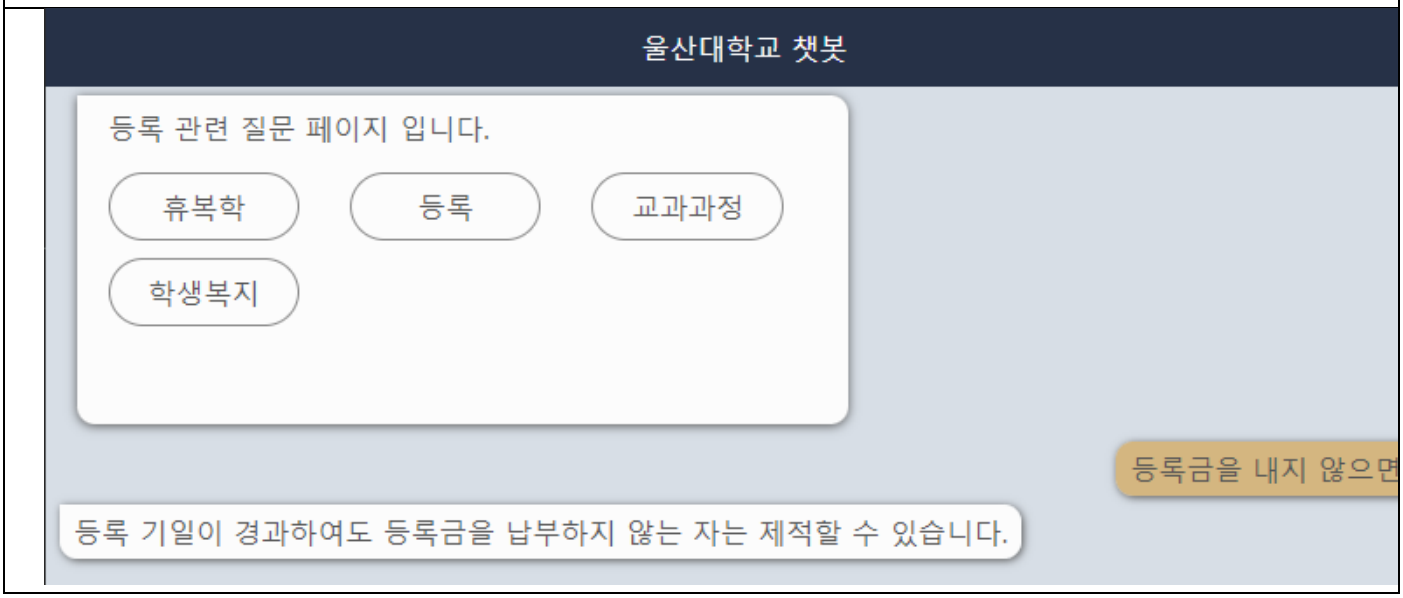
포트폴리오

프로젝트 – 울산대학교 챗봇 시스템 개발

개발 기간	2021.06.30 ~ 2022.06.09
개발 환경	Python3 / hugging face – KoElectra / pycharm – professional -
개발 인원	3명 -> 2명 (도중에 한명 휴학)
요 약	흩어져있는 울산대학교 Q&A를 챗봇을 통해서 한 곳으로 통합
기능	**1) 휴학복학에 관한 Q&A** **2) 등록에 관한 Q&A** **3) 사용자의 질문이 1000개가 차면 자동으로 학습**
실행 화면	





1. 개발 기획

MileStone	21.06	21.07	21.08	21.09	21.10	21.11	21.12	22.01	22.02	22.03	22.04	22.05
분석												
아이디어 회의												
수요 조사												
현재기술 조사												
필요지식 스터디												
설계												
개발 모델 결정												
데이터 확보												
프론트엔드 제작												
백엔드 제작												
AI 모델 설계												
추가 기능 논의												
테스트, 디버그												
배포												

1.1 요구사항 :

기능적 요구사항

- 시스템은 사용자에게 질문을 입력 할 수 있는 영역을 제공
- 시스템은 사용자가 질문을 입력 하면 적절한 대답을 출력
- 사용자가 원하는 대답을 받지 못했을 때 시스템은 다른 답변 또는 솔루션 제시

프로덕트 요구사항 – 유용성

- 사용자에게 질문 입력 완료를 Enter 키 또는 마우스 클릭으로 표현 할 수 있게 제공
- UI의 채팅 박스는 수직적으로 밑으로 계속 쌓음. 이때 화면의 세로 길이는 증가한다
- 사용자의 입력 또는 시스템의 출력시 자동으로 스크롤바가 제일 밑으로 이동

프로덕트 요구사항 – 효율성

- 사용자가 질문 입력 후 답변 받는데 까지의 시간은 2sec 이내

프로덕트 요구사항 – 신뢰성

- 대답가능한 영역을 축소하더라도 답변에 대한 정확도를 높임
- 변동성이 있는 답변은 울산대학교 공식 홈페이지로 연결을 유도하거나 상담원에게 연결을 유도

조직 요구사항 – 배포

- Django를 이용하여 웹페이지를 만든 후 배포 및 서비스
- 배포는 AWS의 EC2를 사용

조직 요구사항 – 구현

- Python을 메인 언어로 사용

- 딥러닝 라이브러리는 pytorch 사용
- Data크롤링은 울산대학교 정보통신원에서 제공 받되 부족한 것은 BeautifulSoup와 Selenium을 혼합해서 해결

1.2 타사 아이템 분석

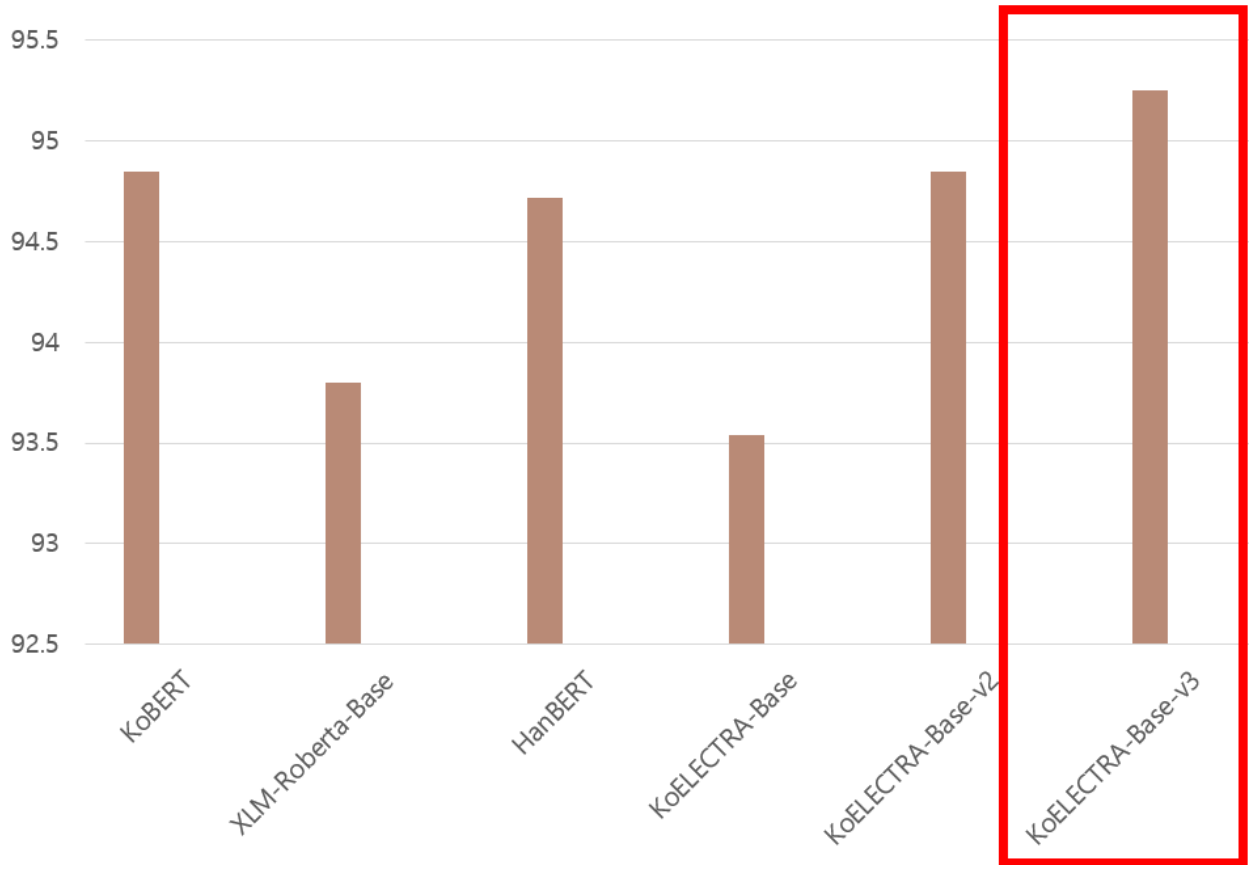
단위 : 1000원

	의뢰 비용	월 지속 비용
채x톡	문의 필요	70
깃x챗	문의 필요	50
단x AI	문의 필요	30
Cloxx xxring	2000	100

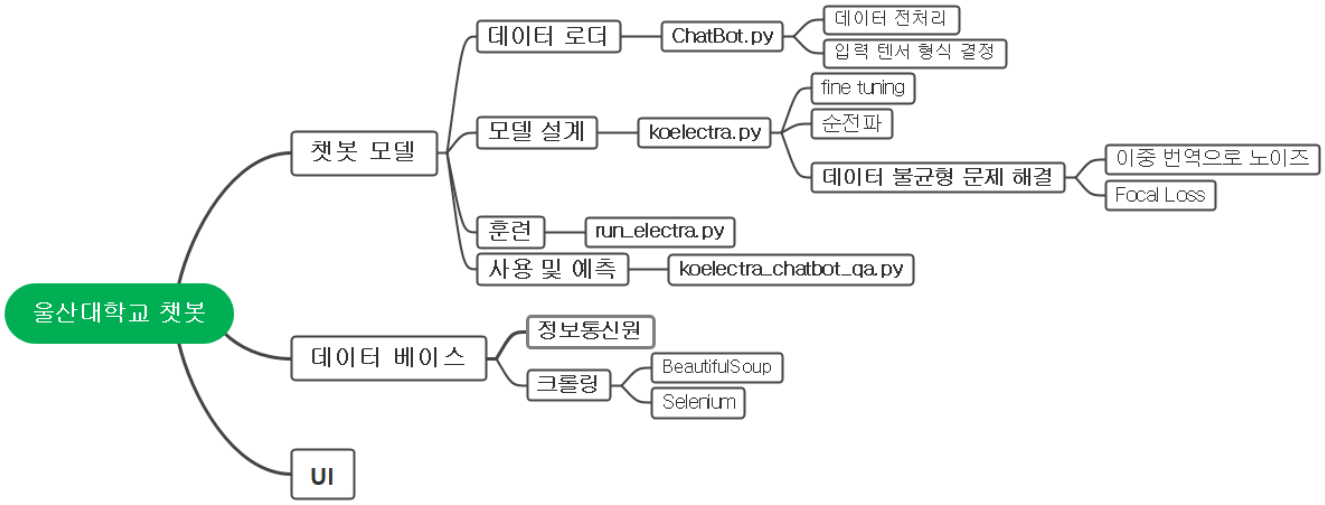
- 초기 설계 비용이 꽤 들어감
- 유지에도 비용이 다소 들어감
- 기존 챗봇 연구는 Bert와 GPT 중심의 연구

차별점 :

- 기존 챗봇은 Bert 모델이 중심이었음 그러나 이 연구에서 사용한 기반기술은 Bert보다 높은 성능을 지니고 있는 최신 기술임
- 학부생이 자체 개발한 기술로서 제작 단가가 저렴
- 데이터가 쌓이면서 유지보수를 하면 무궁무진한 연구 잠재력이 있음



1.3 마인드맵을 이용해 필요한 기능과 로직 분석.



2.1 dataloader.Chatbot.ChatbotTextClassicification 클래스

Code	해설
`index_of_words = self.tokenizer.encode(datas[1])` `token_type_ids = [0] * len(index_of_words)` `attention_mask = [1] * len(index_of_words)`	질문 데이터의 길이를 구하고 그 길이만큼 token_type_ids변수와 attention_mask 변수에 넣음 이 것은 후에 input으로 들어감
`padding_length = max_seq_len - len(index_of_words)` `index_of_words += [0] * padding_length` `token_type_ids += [0] * padding_length` `attention_mask += [0] * padding_length`	Input 문장 데이터의 길이를 모두 동일하게 맞춰주기 위한 과정

2.2 model.koelectra.ElectraClassificationHead 클래스

Code	해설
`class ElectraClassificationHead(nn.Module):` `  """Head for sentence-level classification tasks."""` `  def __init__(self, config, num_labels):` `  super().__init__()` `  self.dense = nn.Linear(config.hidden_size, 4*config.hidden_size)` `  self.dropout = nn.Dropout(config.hidden_dropout_prob)` `  self.out_proj = nn.Linear(4*config.hidden_size, num_labels)` `  def forward(self, features, **kwargs):` `  x = features[:, 0, :]  # take <s> token (equiv. to [CLS])` `  x = self.dropout(x)` `  x = self.dense(x)` `  x = get_activation("gelu")(x)  # although BERT uses tanh here, it s` `  x = self.dropout(x)` `  x = self.out_proj(x)` `  return x`	순전파가 하나 돌아갈때 fine tune 구조. 2개의 dropout layer 계층을 줌으로서 overfitting을 방지하고자 함

2.3 model.koelectra.koElectraForSequenceClassification 클래스

Code	해설
`# 분류 레이블이 2개면 다중 분류 mean square err, 그 이상이면 crossentropy` `  if labels is not None:` `  if self.num_labels == 1:` `  #  We are doing regression` `  loss_fct = MSELoss()` `  loss = loss_fct(logits.view(-1), labels.view(-1))` `  else:` `# crossentropy는 데이터 불균형 문제에서 취약. 논문참조 focal loss` `  #loss_fct = CrossEntropyLoss()` `  #loss = loss_fct(logits.view(-1, self.num_labels), labels.view(-1))` `  loss = focal_loss(logits.view(-1, self.num_labels), labels.view(-1),` `  outputs = (loss,) + outputs`	Loss 값 계산 과정 레이블이 2개면 MSE를 사용 그 이상이면 Focal loss 사용

```python
def koelectra_input(tokenizer, str, device = None, max_seq_len = 512):
  index_of_words = tokenizer.encode(str)
  attention_mask = [1] * len(index_of_words)

  # Padding Length
  padding_length = max_seq_len - len(index_of_words)
  # Zero Padding
  index_of_words += [0] * padding_length
  attention_mask += [0] * padding_length

  data = {\
    'input_ids': torch.tensor([index_of_words]).to(device),
    'attention_mask': torch.tensor([attention_mask]).to(device),
  }
  return data
```

Input 문장을 모델에 집어 넣기 위해 형식을 변경 해주는 함수

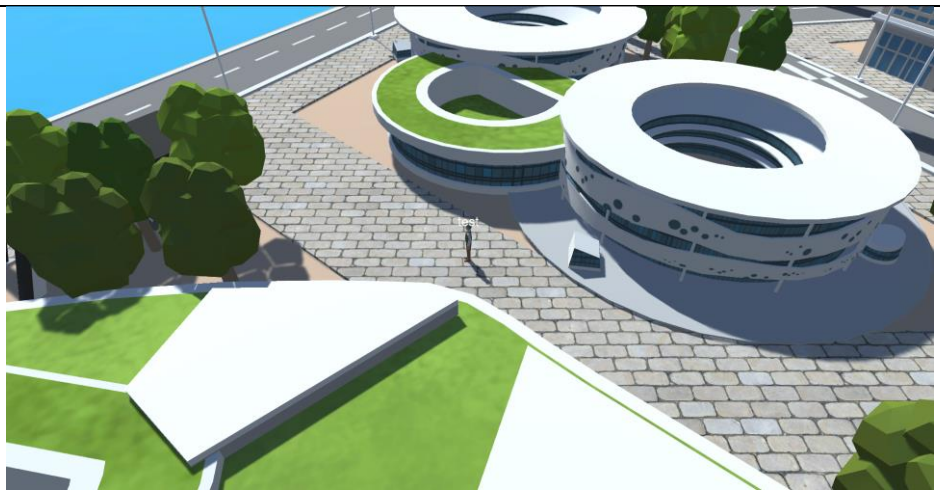
맡은 역할	1. 조장 2. pretrain 모델 코드 분석 3. fine tuning 설계 및 AI모델 제작 4. 챗봇 아이디어 제공 5. 요구사항 설명서 작성 6. 보고서 작성 7. 웹 프로그래밍 8. 데이터 전처리
프로젝트를 진행하면서 어려웠던 점	**1. 팀원 한명의 따라오는 속도가 다소 느렸음.** -> 능력의 문제라 어쩔수 없었기에 빠르게 능력 파악 후 충분히 할 있는 업무를 할당 **2. 팀원 한명이 졸업작품에 투자하는 시간이 매우 적었음.** -> 소통의 부재 또는 의지의 문제라고 판단하였음. 꾸준히 식사 자리와 삶을 물어보면서 소통 길을 열어두고 멘탈을 케어 하고자 하였음. 딱히 다른 이유가 없음에도 불구하고 5개월에 걸친 권유와 기다림과 믿음에도 변화가 없자 강력하게 경고 . **3. 팀원들의 기여도가 줄어들수록 나의 일이 너무 많아짐** -> 버티고 버텼음. 그리고 1년이 지난 지금 인성과 실력면에서 많이 성장하였음. **4. 데이터 불균형 문제** -> 레이블간의 데이터의 개수가 차이가 나서 pytorch라이브러리에서 제공하는 cross entropy 함수로만 계산하면 개수가 적은 레이블의 데이터는 지나치게 적게 훈련이 되어 정확도가 떨어지는 현상이 나타났음. cross entropy의 함수를 변형시켜서 만든 Focal Loss를 자체적으로 작성하여 개수가 적은 레이블에 대해서 가중치를 많이 줌으로서 해결함

깨달은 점	1. 혼자서는 절대 양질의 프로젝트를 완성시킬 수 없음 2. 프로젝트에서 가장 중요한 것은 협업 3. 사업 아이디어 내는 것이 정말 힘듦 4. 듣기 싫은 소리도 리더로서 해야 할 때가 있음 5. 발표자료는 측량 가능한 수치로 할 것 6. 보고서 쓸 때 단위 표기가 매우 중요함

프로젝트 – 유니티 메타버스 시스템 개발

개발 기간	2022.03.23 ~ 2022.05.08
개발 환경	Unity 2020.03.33 / VisualStudio 2019 / C#
개발 인원	3명
요 약	웹기반 메타버스 플랫폼 운영
기능	**1. 임의 위치 동영상 재생** **2. 임의 위치 유튜브 링크 재생** **3. 유니티 내 웹페이지 연동** **4. SNS 연동** **5. 방명록 구현** **6. 게시판 구현** **7. 채팅 구현 (텍스트, 영상)** **8. 아바타 기본 동작 (걷기, 손흔들기, V포즈, 손흔들기)** **9. 특정 아바타랑 나의 아바타랑 사진 찍기**

실행 화면



1. 개발 기획

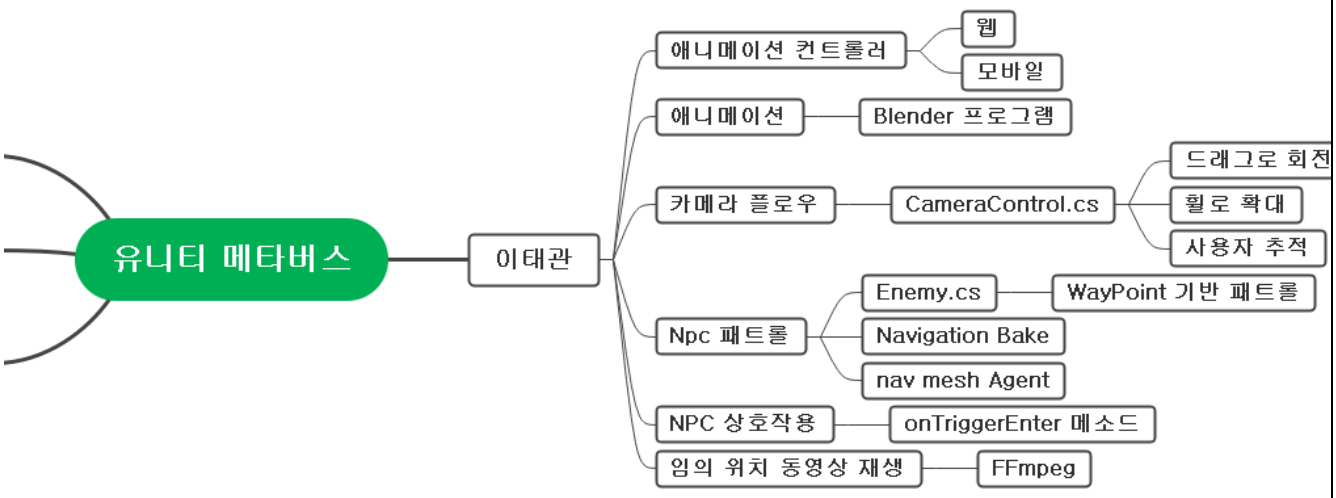
MileStone	03.23	03.30	04.06	04.13	04.20	04.27	05.04	05.09	05.16	05.23	05.30	06.06

기능구현												
동영상 재생												
유튜브 재생												
웹페이지 연동												
SNS 연동												
방명록												
게시판												
채팅												
아바타												
애니메이팅												
상호작용												
카메라 플로우												
Npc 패트롤												
운영테스트												
부하 테스트												
추가 수정 작업												

1.1 요구사항 : (전 말단이라 정확하게는 잘 모릅니다)

1. 디자인을 제외한 흔한 쿼터뷰 게임에서 볼 수 있는 기능들
2. 웹 기반 메타버스 플랫폼 운영
3. Unity 기반 메타버스 플랫폼 구축
4. 메타버스 내 아바타 및 물체 컨트롤 기능 구현
5. 전광판 내 영상 재생(스트리밍도 되게)
6. 채팅 기능 구현
7. 응원 게시판 구현

1.3 마인드맵을 이용해 필요한 기능과 로직 분석.



2.1 CameraControl 클래스 : 카메라의 줌, 휠업시의 확대 휠 다운시 축소 및 카메라 회전 역할

Code	해설
`` // by태관 카메라 줌. 휠업시 확대 휠 다운시 축소 void Zoom() { if (Input.GetAxis("Mouse ScrollWheel") != 0) { Distance += Input.GetAxis("Mouse ScrollWheel") * ZoomSpeed * -1; AxisVec = transform.forward * -1; if (Input.GetAxis("Mouse ScrollWheel") > 0) { if (Distance <= 9.8f) { Distance = 9.8f; } else { AxisVec += (Distance * -1); transform.position = MainCamera.position + AxisVec; } } else { if (Distance >= 10.1f) { Distance = 10.1f; } else { AxisVec += Distance; transform.position = MainCamera.position + AxisVec; } } } } ```	마우스 휠을 돌려서 위로 돌렸을때는 확대(Distance를 줄이기)하고 아래로 돌리면 축소(Distance를 늘리기) 하는 기능. 최대확대(최소 Distance)와 최소 축소(최대 Distance)는 각각 9.8, 10.1로 설정 해놓은 모습

```
// by태환 카메라 회전.
void CameraRotation()
{
    if (transform.rotation != TargetRotation)
        transform.rotation = Quaternion.Slerp(transform.rotation, TargetRotation, RotationSpeed * Time.deltaTime);

    if (Input.GetMouseButton(1))
    {
        // 값을 축적.
        Gap.x += Input.GetAxis("Mouse Y") * RotationSpeed * -1;
        Gap.y += Input.GetAxis("Mouse X") * RotationSpeed;

        // 카메라 회전범위 제한.
        Gap.x = Mathf.Clamp(Gap.x, -5f, 85f);
        // 회전 값을 변수에 저장.
        TargetRotation = Quaternion.Euler(Gap);

        // 카메라벡터 객체에 Axis객체의 x,z회전 값을 제외한 y값만을 넘긴다.
        Quaternion q = TargetRotation;
        q.x = q.z = 0;
        CameraVector.transform.rotation = q;
    }
}
```

쿼터뷰 게임의 회전을 구현

마우스 오른쪽 벝튼을 누르면 회전 쿼터니언 값을 받아와서 회전 후 카메라는 target(Player)를 정 중앙에 오게끔 다시 움직인다

2.2 Player 클래스 : Player의 움직임과 애니메이션컨트롤러의 연동

Code	해설
`void Move()` `{` `  moveVec = new Vector3(hAxis, 0, vAxis).normalized;` `  if (isDodge)` `  {` `  moveVec = dodgeVec;` `  }` `  if (wDown)` `  {` `  transform.position += moveVec * speed * 0.3f * Time.deltaTime;` `  }` `  else` `  {` `  transform.position += moveVec * speed * Time.deltaTime;` `  }` `  anim.SetBool("isRun", moveVec != Vector3.zero);` `  anim.SetBool("isWalk", wDown);` `}`	캐릭터가 움직이는 것을 구현. wDown은 걷기임
`void Jump()` `{` `  if (jDown && !isJump && moveVec == Vector3.zero)` `  {` `  rigid.AddForce(Vector3.up * 15, ForceMode.Impulse);` `  isJump = true;` `  anim.SetBool("isJump", true);` `  anim.SetTrigger("doJump");` `  }` `}`	캐릭터가 점프 하는 것을 구현. 두번째 메소드는 캐릭터가 바닥과 닿았을때 애니메이션 컨트롤러에게 '점프 끝났다' 라고 알려줌

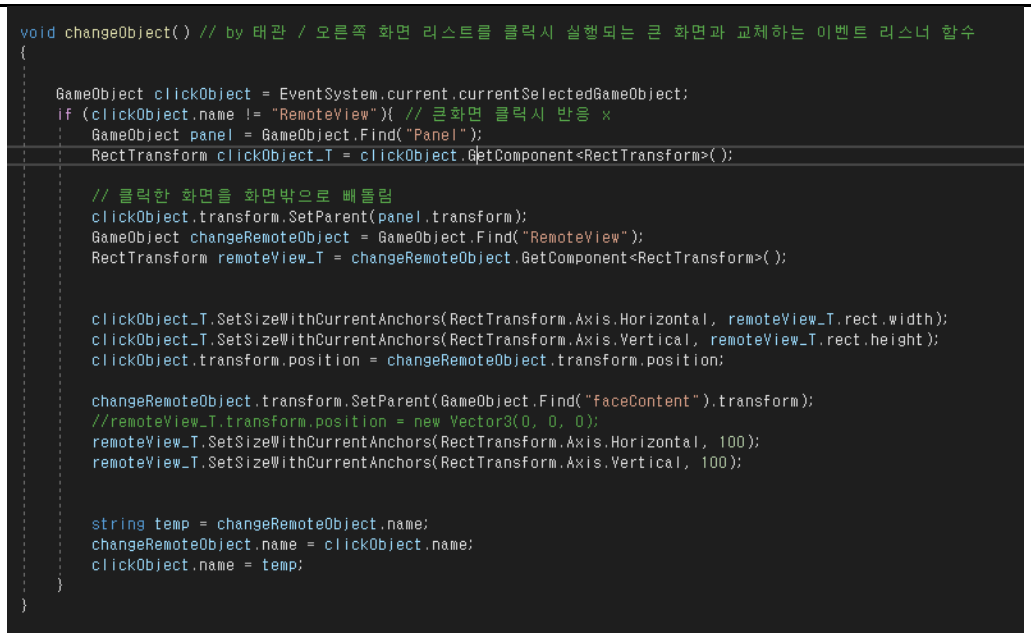
```
void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.tag == "Floor")
    {
        anim.SetBool("isJump", false);
        isJump = false;
    }
}
```

```
void Dodge()
{
    if (jDown && !isJump && moveVec != Vector3.zero && !isDodge)
    {
        dodgeVec = moveVec;
        speed *= 2;
        anim.SetTrigger("doDodge");
        isDodge = true;

        Invoke("DodgeOut", 0.4f);
    }
}

void DodgeOut()
{
    speed *= 0.5f;
    isDodge = false;
}
```

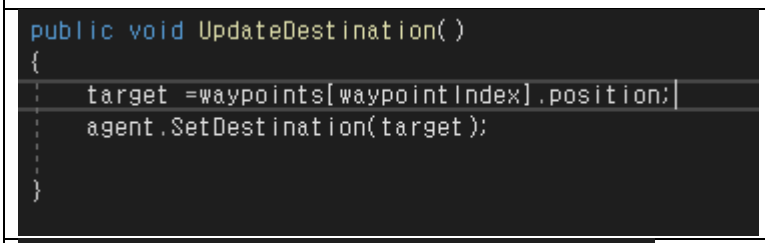
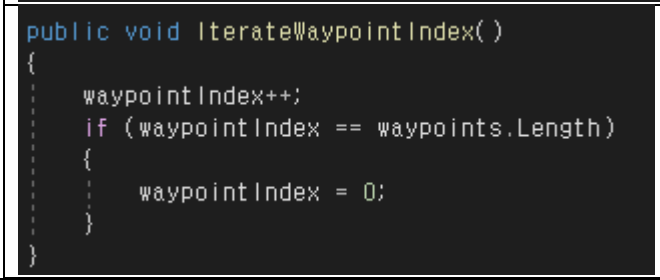
캐릭터가 대쉬를 했 했을 때 구현.

2.3 GameManager 클래스 : Agora asset 사용하여 유니티 내 영상통화 기능

Code	해설
`void onClickAdd() // by태관  참여자가 추가됐을때 화면 오른쪽 리스트에 추가 시키는 함수` `{` `GameObject temp = Instantiate(test, new Vector3(0, 0, 0), Quaternion.identity);` `temp.name = "user" + userCount.ToString();` `userCount++;` `temp?.GetComponent<Button>()?.onClick.AddListener(changeObject);` `GameObject faceContent = GameObject.Find("faceContent");` `temp.transform.SetParent(faceContent.transform);` `}`	1:1의 상황일때는 상대방의 화면을 큰 화면에 위치. 후에 추가로 사람이 추가되면 오른쪽 리스트 뷰에 작은 화면에 상대방들의 화면을 위치 temp객체에 user(숫자) 이름을 붙이고 faceContent 객체 밑으로 옮김

	오른쪽 리스트뷰 작은 화면을 클릭하면 그 작은 화면이 큰 화면으로 옮겨지고 큰 화면은 작은 화면으로 옮겨짐

2.4 Enemy 클래스 : 자동차, 사람, 세그웨이 패트롤 기능

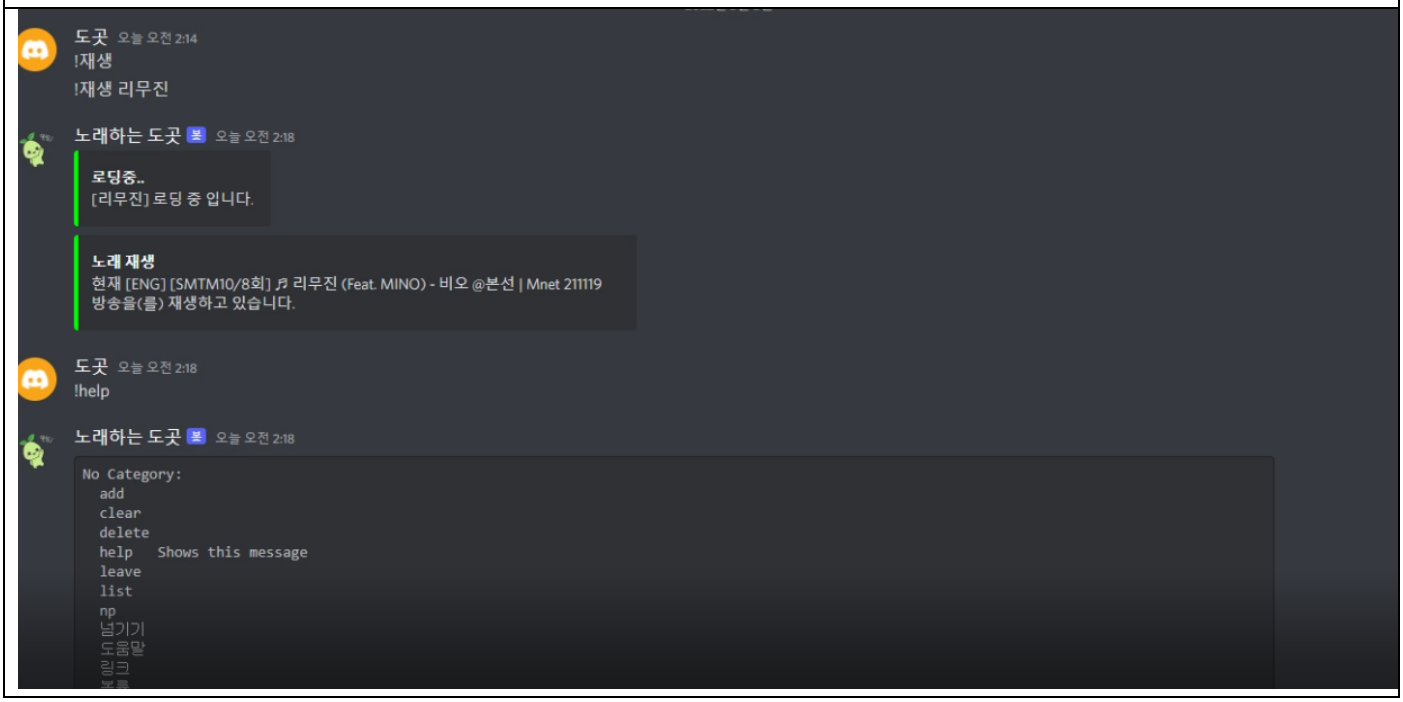
Code	해설
void Update() { //Debug.Log(target); if (Vector2.Distance(new Vector2(transform.position.x,transform.position.z), new Vector2(target.x, target.z)) < 1) { IterateWaypointIndex(); UpdateDestination(); } }	WayPoint와 가까워졌을 때 ItererateWayPointIndex()를 호출 하여 다음 waypoint를 찾고 UpdateDestination()을 호출 하여 다음 waypoint로 이동한다
	Waypoint의 해당 인덱스로 이동한다
	Waypoint 배열의 다음 index를 가져온다

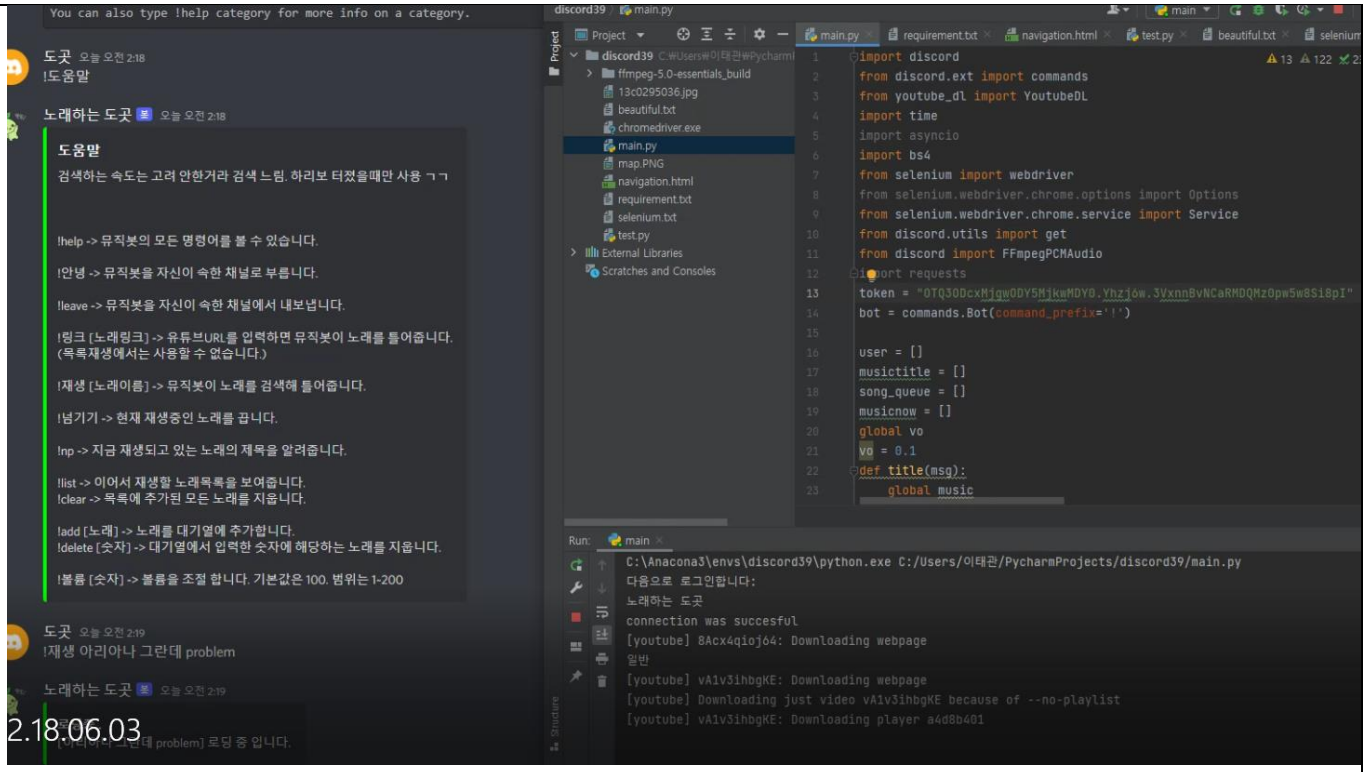
맡은 역할	1. 대량 트래픽을 한번에 감당할수 있는 서버 – 클라이언트 구조 개발 2. 애니메이션 컨트롤러 로직 설계 3. 간단한 애니메이션 제작 4. 쿼터뷰 게임에서 볼 수 있는 카메라 플로우 로직 설계 5. 자동차와 세그웨이, 지하철의 패트롤 설계 6. npc와 부딪혔을때의 상호작용 개발 7. 전광판에서 동영상 재생 8. Agora asset 을 사용한 영상채팅.
프로젝트를 진행하면서 어려웠던 점	**1. photon asset을 사용하면 한 채널에 최대 16명까지 수용 가능하지만 쉽게 구현이 가능하고 연구실 서버 컴퓨터를 활용해서 자체적으로 서버를 돌릴 수 만 있다면 100명이고 200명이고 수용이 가능하다고 교수님이 그렇게 해보라고 요구 하셔서 이틀 정도 집에도 못들어가고 연구실에서 하루종일 해커톤을 했음. 이틀째 밤새고 다음날 오전 9시 교수님 출근하시고 나서 그냥 하던거 폐기하고 photon으로 쉽게 쉽게 가자고 하셨음. 이틀의 노력이 물거품이 되어서 멘탈적으로 힘들었지만 그래도 어디가서 1억짜리 자체 서버 컴퓨터로 서버개발 처음부터 할 기회 없을텐데 귀한 경험했다고 자기암시를 하며 멘탈을 지켜내며 극복.** 2. 우리는 메타버스 플랫폼을 개발만 하면 되었음. 메타버스 장소와 사람 디자인과 애니메이션은 모두 다른 회사가 만들어서 전달해주기로 했는데 너무 늦게 4월 28일에 전달받았음. 때문에 개발일정이 배로 빠듯해졌었음 3. 미리 요구를 하지못한 간단한 애니메이션 (손흔들기, v사진포즈)들이 필요했었는데 그나마 할일이 적었던 내가 구현했음. 아예 생전 처음보는 툴을 갑작스럽게 공부해서 성과를 내야하는게 부담되었지만 진득하게 의자에 하루종일 앉아서 공부함으로서 극복 .
깨달은 점	**1. 개발 계획대로 착착 진행되는 경우는 거의 없다** **2. 연구개발실 분위기 축 처지게 안되게 만들고 집중할 수 있는 환경을 만드는 것도 리더의 능력이다. 그래서 팀장님이 존경스러웠다.** **3. 실력 향상엔 모르더라도 일단 일이 맡겨지면 코피터져가면서 데드라인 지키기위해 발버둥 치는 것이 최고다** **4. 작업을 의뢰한 사람은 본인이 무엇을 원하는지 잘 모른다** **5. 뭘 물어볼땐 바로 앞기수 대학원생에게 물어봐야한다** **6. 개발과 야근은 관련성이 많다. 미리 마음의 준비를 해야한다** **7. 라이브 서비스 할 때 가장 중요한 것은 안정성**

여기부턴 미니프로젝트 또는 토이프로젝트 입니다

미니프로젝트 – 디스코드 음악 봇 개발

개발 기간	2022.02.24 ~ 2022.02.26
개발 환경	Python3 / pycharm – professtiona;
개발 인원	1명
요 약	디스코드 음악 봇 개발
기능	**1. 음악 재생** **2. 음악 리스트 보관 후 재생** **3. 다음 곡으로 넘기기** **4. 볼륨조절** **5. 음악 리스트 index 삭제** **6. 음악 리스트 전체 삭제**
실행 화면	





1.1 요구사항

1. 기존에 쓰던 노래하는 하리보라는 디코봇이 서버 문제로 너무 렉이 걸리므로 렉 안걸리게
2. 노래하는 하리보와 똑같거나 비슷한 커맨드
3. 노래 재생, 볼륨 조절, 넘기기 기능 필수

1.2 왜 필요한가?

1. 상용 디코 음악 봇은 공용서버가 과부하가 걸리면 지나치게 느려짐

2.1 재생 메소드 : 음악 재생 기능

Code	해설
async def 재생(ctx, *, msg): emb = discord.Embed(color = 0x00ff00) emb.add_field(name="로딩중..", value="[" + msg + "] 로딩 중 입니다.") await ctx.send(embed=emb) try: global vc vc = await ctx.message.author.voice.channel.connect() except: try: channel_name = ctx.message.author.voice.channel print(channel_name) if channel_name == "방역준수채널": await vc.move_to(channel_name) except: await ctx.send("채널에 유저가 접속해 있지 않아요") if not vc.is_playing():	사용자의 msg를 불러와서 봇을 현재 채널로 불러들인다.

`` if not vc.is_playing(): options = webdriver.ChromeOptions() options.add_argument("headless") global entireText YDL_OPTIONS = {'format': 'bestaudio', 'oplaylist': 'True'} FFMPEG_OPTIONS = {'before_options': '-reconnect 1 -reconnect_streamed 1 -reconnect_delay_max 5', 'options': '-vn'} chromedriver_dir = "chromedriver.exe" service = Service(chromedriver_dir) driver = webdriver.Chrome(service=service, options=options) driver.implicitly_wait(3) ul = "https://www.youtube.com/results?search_query=" + msg driver.get(ul) time.sleep(0.5) source = driver.page_source bs = bs4.BeautifulSoup(source, 'lxml') entire = bs.find_all('a', {'id': 'video-title'}) entireNum = entire[0] entireText = entireNum.text.strip() musicurl = entireNum.get('href') url = 'https://www.youtube.com' + musicurl driver.quit() musicnow.insert(0,entireText) with YoutubeDL(YDL_OPTIONS) as ydl: info = ydl.extract_info(url, download=False) URL = info['formats'][0]['url'] vc.play(FFmpegPCMAudio(URL, **FFMPEG_OPTIONS, executable="D:/Download/ffmpeg-5.0-essentials_build/ffmpeg-5.0-essentials_ after=lambda e: play_next(ctx)) vc.source = discord.PCMVolumeTransformer(vc.source) vc.source.volume = vo emb = discord.Embed(color = 0x00ff00) emb = emb.add_field(name="노래 재생", value="현재 " + musicnow[0] + "을(를) 재생하고 있습니다.") await ctx.send(embed=emb) ```	만약 봇이 실행되고 있지 않다면, Selenium으로 유튜브에서 사용자 msg로 검색한다. 가장 위에 있는 동영상을 다운 받고 인코딩을 시켜준뒤 실행시킨다

2.2 np 메소드 : 현재 재생 중인 음악 이름 구하기 기능

Code	해설
`` @bot.command() async def np(ctx): if not vc.is_playing(): await ctx.send("지금은 노래가 재생되지 않네요..") else: await ctx.send(embed = discord.Embed(title = "지금노래", description = "현재 " + musicnow[0] + "을(를) 재생하고 있습니다.", color = 0x00ff00)) ```	지금 봇이 재생 되고 있다면 musicnow 인덱스의 첫 번째 곡을 사용자에게 알려줌

2.3 넘기기 메소드 : 다음 곡 전환 기능

Code	해설
`@bot.command()` `async def 넘기기(ctx):` `  if vc.is_playing():` `  vc.stop()` `  await ctx.send(embed = discord.Embed(title= "노래끄기", description = musicnow[0] + "을(를) 종료했습니다.", colon = 0x00ff00))` `  else:` `  await ctx.send("지금 노래가 재생되지 않네요.")`	현재 노래 종료
`def play_next(ctx):` `  if len(musicnow) - len(user) >= 2:` `  for i in range(len(musicnow) - len(user) - 1):` `  del musicnow[0]` `  YDL_OPTIONS = {'format': 'bestaudio', 'noplaylist':'True'}` `  FFMPEG_OPTIONS = {'before_options': '-reconnect 1 -reconnect_streamed 1 -reconnect_delay_max 5', 'options': '-vn'}` `  if len(user) >= 1:` `  if not vc.is_playing():` `  del musicnow[0]` `  URL = song_queue[0]` `  del user[0]` `  del musictitle[0]` `  del song_queue[0]` `  vc.play(discord.FFmpegPCMAudio(URL,**FFMPEG_OPTIONS,executable="D:/Download/ffmpeg-5.0-essentials_build/ffmpeg-5` `  vc.source = discord.PCMVolumeTransformer(vc.source)` `  vc.source.volume = vo` `  print(vc.source.volume)`	종료 했는데 다음 곡이 있을땐 음악 큐의 0을 없애고 새로 들어온 음악큐의 0 인덱스를 실행 시킴

2.4 볼륨 메소드 : 볼륨 조절 기능

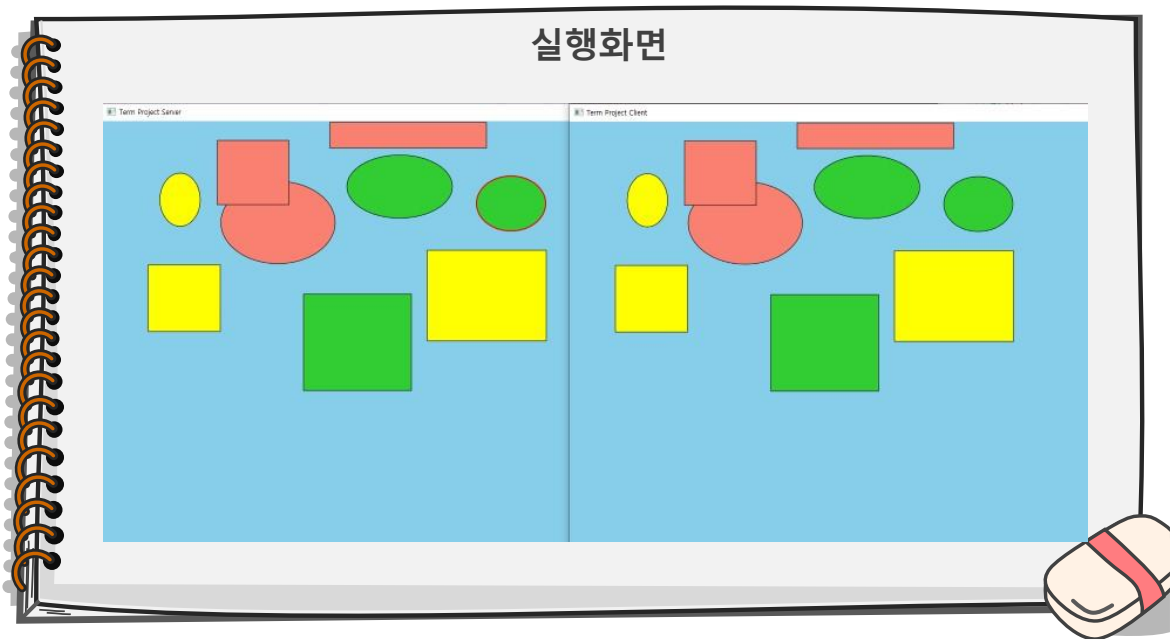
Code	해설
`@bot.command()` `async def 볼륨(ctx, *, vol):` `  if float(vol)>0 and float(vol)<=200:` `  try:` `  await vc.move_to(ctx.message.author.voice.channel)` `  try:` `  before = vc.source.volume` `  vc.source.volume = float(vol)/1000` `  global vo` `  vo = float(vol)/1000` `  await ctx.send("볼륨이 " + str(before*1000) + "에서 " + str(vo*1000) +"으로 변경 되었어요.\n기본값은 100")` `  except:` `  await ctx.send("노래가 실행 중이지 않아요")` `  except:` `  await ctx.send("유저가 음성 채널에 접속해 있지 않아요")` `  else:` `  await ctx.send("볼륨 조절 범위를 확인해 주세요 (1~200)")`	!볼륨 110을 하면 110의 숫자를 받아서 디스코드 볼륨을 110으로 바꾼다. 최소 0 최대 200 기본 100

프로젝트를 진행하면서 어려웠던 점	1. discord.py 의 api Document만 보고 개발하였는데 너무 난잡하게 되어있어서 찾는데 시간이 오래 걸림
깨달은 점	**1. 자체 서버를 돌리면 개발하는데 피곤하긴 하지만, 한번 개발해놓으면 나중이 편하다**

미니프로젝트 – TCP 소켓 통신으로 server – client 구조의 도형 드리기

개발 기간	2021.10.31 ~ 2021.12.16
개발 환경	C++ / Visual Studio 2019
개발 인원	1명
요 약	TCP 프로토콜의 소켓 통신에 대한 이해
기능	**1. client가 그림을 그리면 server가 좌표값 얻어와서 server 컴퓨터에 그림**

실행 화면



1.1 요구사항

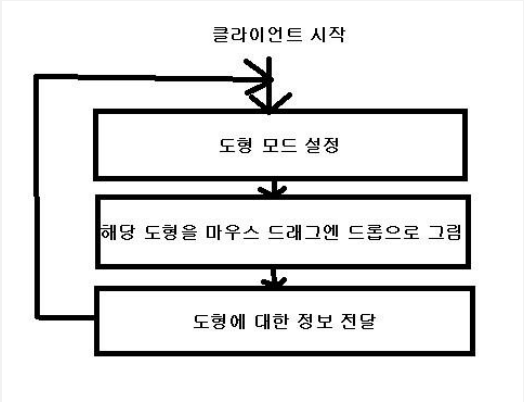
1. 도형 그리는데 라이브러리 쓰면 안됨
2. 통신에서 오고 가는 것은 도형의 좌표만 허용
3. TCP, UDP 뭐로 하든 상관 없음

요구사항 설명서-사용자 요구사항

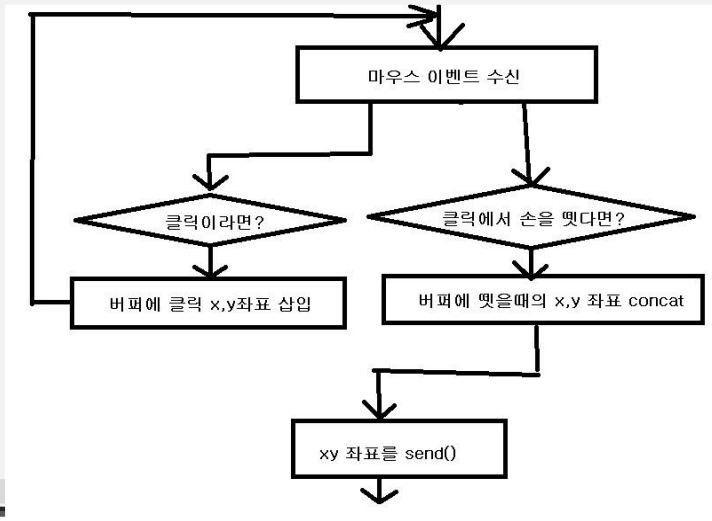
1. SimpleDrawing_Server.exe 실행 (SimpleDrawing_Server.sln 컴파일)
2. SimpleDrawing_Client.exe 실행 (SimpleDrawing_Client.sln 컴파일)
3. F1키는 타원 그리기로 전환 F2는 직사각형 그리기로 전환
4. 마우스 드래그로 도형을 그림

클라이언트의 동작과정

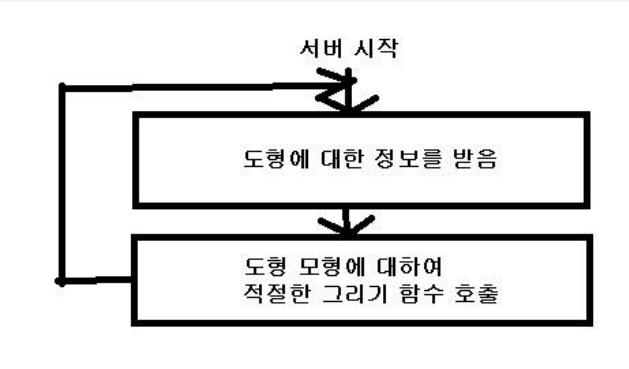
• 참고 코드의 Microsoft 예제 자료에서
모드를 변경 하는 기능으로
도형 잡기, 삭제 등을 구현했는데
이걸 활용 하여 도형을
선택 하는 것으로 구현 하겠다.



도형에 대한 정보 전달 : 클라이언트



서버의 동작과정

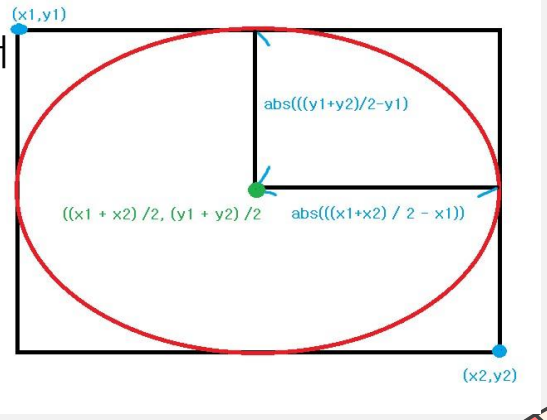


직사각형 그리기 원리

• 직사각형은
마우스 이벤트로 얻을 수 있는
LEFT(x1) TOP(y1)
RIGHT(x2) BOTTOM(y2)점들을
단순히 잇는 과정을 통해 그린다

원 그리기 원리

• 원은 마우스 클릭시의 좌표와
클릭 해제시의 좌표를 이용해서
원의 중심을 구하고
긴 반지름과 짧은 반지름을
계산 한 후 원을 그린다



2.1 : SimpleDrawingClient.sln 코드

Code	해설
`return 0;` `// 마우스 클릭 다운의 x,y좌표 저장` `case WM_LBUTTONDOWN:` `OnLButtonDown(GET_X_LPARAM(lParam), GET_Y_LPARAM(lParam), (DWORD)wParam);` `itoa(GET_X_LPARAM(lParam), first_buf, 10);` `strcat_s(first_buf, " ");` `itoa(GET_Y_LPARAM(lParam), &first_buf[strlen(first_buf)], 10);` `len = strlen(first_buf);` `strcat_s(first_buf, " ");` `return 0;` `// 마우스 클릭 업의 x,y좌표 저장` `case WM_LBUTTONUP:` `OnLButtonUp();` `itoa(GET_X_LPARAM(lParam), second_buf, 10);` `strcat_s(second_buf, " ");` `itoa(GET_Y_LPARAM(lParam), a, 10);` `strcat_s(second_buf, a);` `strcat_s(first_buf, second_buf);` `len = strlen(first_buf);` `if (mode == circleMode) {` `strcat_s(first_buf, " 1");` `}` `else if (mode == squareMode) {` `strcat_s(first_buf, " 2");` `}` `len = strlen(first_buf);`	마우스 클릭 이벤트 발생시 클릭과 해제의 좌표를 각각 구해서 저장한다

`` // 데이터 보내기(고정 길이) retval = send(sock, (char*)&len, sizeof(int), 0); if (retval == SOCKET_ERROR) { err_display("send()"); break; } // 데이터 보내기(가변 길이) retval = send(sock, first_buf, len, 0); if (retval == SOCKET_ERROR) { err_display("send()"); break; } return 0; strcpy_s(first_buf,""); ```	먼저 데이터의 길이를 고정길이로 보낸 후 그 길이만큼 가변길이로 데이터를 보낸다

2.2 SimpleDrawingServer.sln 코드 : 현재 재생 중인 음악 이름 구하기 기능

Code	해설
`` // 만약 데이터를 성공적으로 받았을 경우 if (success) { strtok_s(first_buf, " ", &gived_data); x1 = atoi(first_buf); strtok_s(gived_data, " ", &str1); y1 = atoi(gived_data); strtok_s(str1, " ", &gived_data); x2 = atoi(str1); strtok_s(gived_data, " ", &str1); y2 = atoi(gived_data); shape = atoi(str1); // 타원 그리기 모드 일때 타원을 그리고 if (shape == 1) { InsertEllipse((x1 + x2) / 2, (y1 + y2) / 2, (fabsf(((x1 + x2) / 2) - x1)), (fabsf(((y1 + y2) / 2) - y1))); } // 직사각형 그리기 모드 일때 직사각형을 그린다ㅎ else if (shape == 2) { InsertRect(x1, y1, x2, y2); } str1 = NULL; gived_data = NULL; success = false; receive = true; } return DefWindowProc(m_hwnd, uMsg, wParam, lParam); ```	만약 데이터를 성공적으로 받았을 경우 모드에 따라 직사각형 또는 타원을 그린다.

프로젝트를 진행하면서 어려웠던 점	1. 총 기간은 여유 있었는데 과제와 시험기간 준비한다고 실제 작업시간이 턱없이 부족했음
깨달은 점	**1. 미루면 큰일난다** **2. 앞으로 이런 TCP 환경에서 소켓이 오고가는 개발을 많이 할건데 지금 한 기본기를 잘 닦아 놔야겠다**