Portfolio

Park YeHun

void **Contents**()

```
switch( start )
{
case 1 :
    //   Park Yehun
case 2 :
    //   Status
case 3 :
    //   Projects
case 4 :
    //   Co-work,
    //   Management
default :
    break;
}
```

Lv.1 - 팅!통!팅!(2학년 || 2014)

Lv.2 - 복수의 원(3학년 || 2016)

Lv.3 - 뻐끔뻐끔(4학년 || 2017)

case 1: Park Yehun

E-mail : cru6548@gmail.com

Git hub : https://github.com/gangjung

Other page

+ 공모전 팀
  - https://cru6548.wixsite.com/beyond-imagination

Comment

+ **'F4'를 위하여!!!** // (For Fun, Form Fun)
  - 재밌는 세상을 위해, 재밌는 세상을 만들자!

History & Career

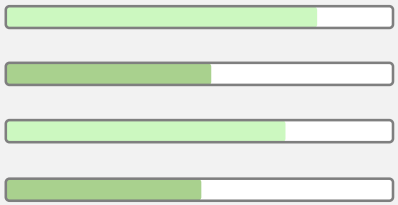
+ 고려대학교 입학(컴퓨터정보) // 2011.03
+ 워킹홀리데이(호주) // 2015.06 ~ 2016.05
+ 교내 *PL센터 학부 도우미(대표) // 2016.09 ~ 2017.12
  (Programing Language Help Center)
+ 과내 게임대회 주최(대표) // 2016.10 ~ 2016.11
+ 팀 구성 및 공모전 참가 // 2017.03 ~
+ 과내 알고리즘 대회(Kuding) "2등" 수상 // 2017.04
+ 공개 소프트웨어 개발자대회 "동상" 수상 // 2017.11.24
+ 한이음 공모전 "입선" // 2017.11.24
+ 고려대학교 CK 캡스톤 디자인 "대상" // 2017.11.24

case 2: **Status**

Class : Computer Science

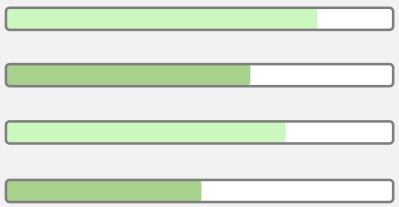
<Computer Language>

>> C / C++
>> C#
>> STL
>> Java



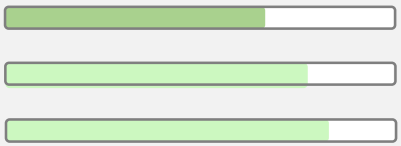
<Major>

>> Algorism
>> Android
>> Unity
>> Blender 3D



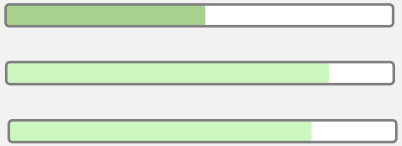
<Personal>

>> English
>> Memo(EverNote)
>> Googling



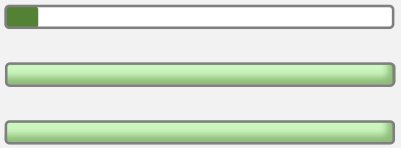
<Project Co-work>

>> Git
>> Documentation
>> Discussion



<Special>

>> Game Control
>> Find Fun
>> Positive mind



case 3: Projects

Lv.1 - 팅!통!팅! (2학년)

+ 개요
// 마우스로 벽을 만들어 공을 목적지까지 보내야 하는 게임

+ 배경
// "윈도우즈 프로그래밍" 과목 프로젝트로, 간단한 충돌처리를 구현.

+ 장르
// 아케이드

+ 개발 인원
// 1인 개발

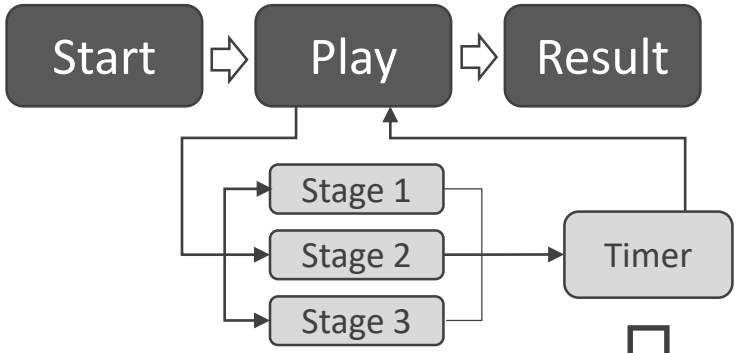
+ 사용언어 및 라이브러리
// 언어 : C
// 라이브러리 : Win API, Fmod

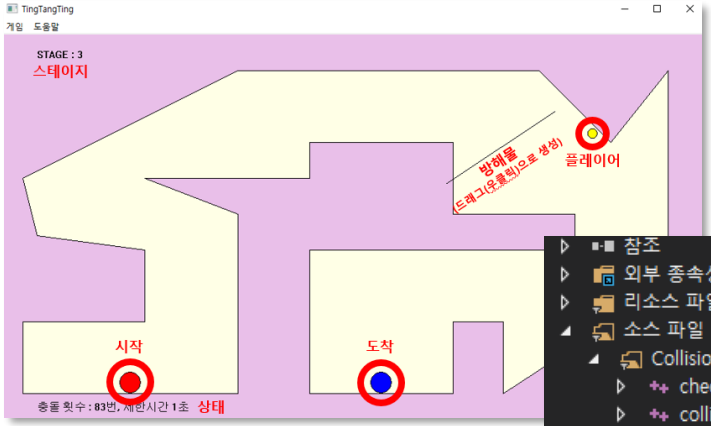
+ 개발 기간
// 2014.05 ~ 06(약 2주)

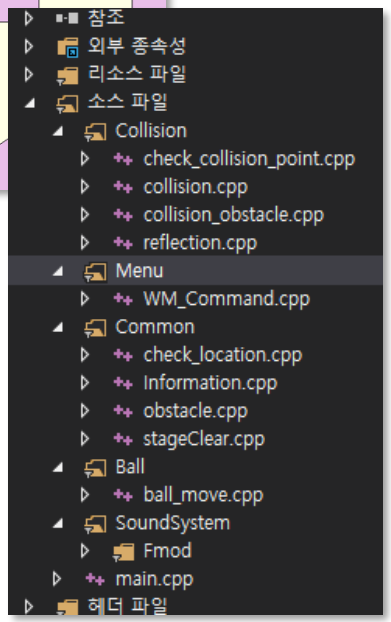
+ 링크
// **[소스코드]** : https://github.com/gangjung/TingTongTing

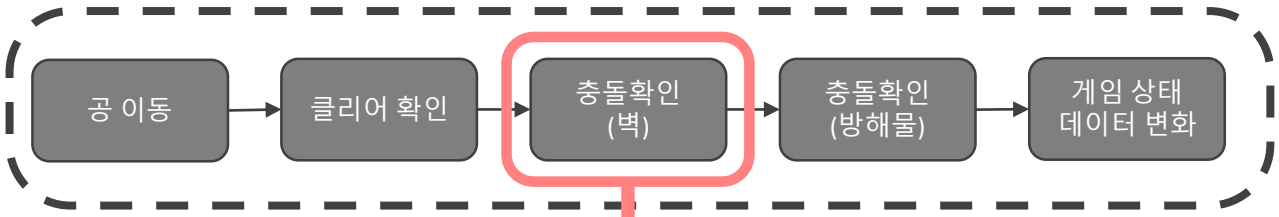
case 3: Projects

전체 로직

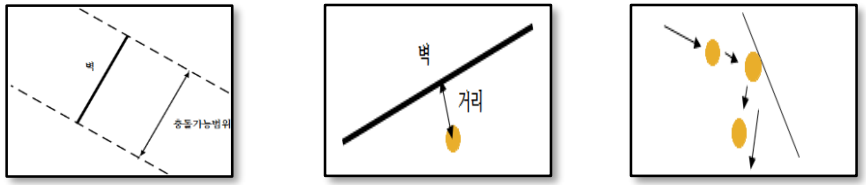






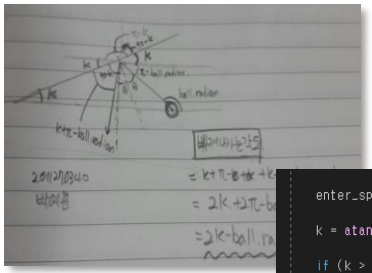


충돌 범위 확인 -> 충돌 확인 -> 반사



case 3: Projects

주요 로직(충돌)



1) check_range(...)
- 충돌 가능한 범위인지 미리 확인하여, 불필요한 연산 제거.

2) check_contact(...)
- 점과 선 사이의 거리를 이용, 충돌 확인.

3) reflection(...)
- 삼각함수를 이용, 벽의 기울기와 입사각을 이용하여 반사각 적용.

```
    enter_speed = ball.speed;

    k = atan2(gradient,1);

    if (k > PI) // 기울기이기때문에 180도보다 클 수 없다.
        k -= PI;
    else if (k < 0)
        k += PI;

    ball.radian = 2 * k - ball.radian;

    if (object == WALL) // 1이면 벽과의 충돌
        ball.reflection = true;
    else if (object == OBSTACLE)    // 2이면 방해물과의 충돌.
        ball.reflection_obstacle = true;
}
else
{
    if (point2.x == point1.x)   // y축과 평행한 선. x속도 방향만 바꿔주면 된다.
    {
        ball.radian = PI - ball.radian;
        if (object == WALL)
            ball.reflection = true;
        else if (object == OBSTACLE)
            ball.reflection_obstacle = true;
    }
    else                        // x축과 평행한 선. y속도 방향만 바꿔주면 된다.
    {
        ball.radian -= ball.radian * 2;
        if (object == WALL)
            ball.reflection = true;
        else if (object == OBSTACLE)
            ball.reflection_obstacle = true;
    }
```

더블 버퍼링

+ 매 Timer마다 WM_PAINT가 실행되는데, 이전의 데이터가 모두 지워지고 다시 그려지기 때문에 '깜빡임 현상'이 발생.

+ 이전의 데이터를 저장하고, 새로 그려지는 화면 뒤에 미리 생성. 그 위에 새로운 데이터를 덧 입혀, 깜빡임 현상 제거.

```
case WM_PAINT:
    hdc = BeginPaint(hwnd, &ps);
    memdc1 = CreateCompatibleDC(hdc);    // 더블버퍼링 사용.
    backBitmap = CreateCompatibleBitmap(hdc, rect.right, rect.bottom);
    SelectObject(memdc1, backBitmap);
    memdc2 = CreateCompatibleDC(hdc);
```

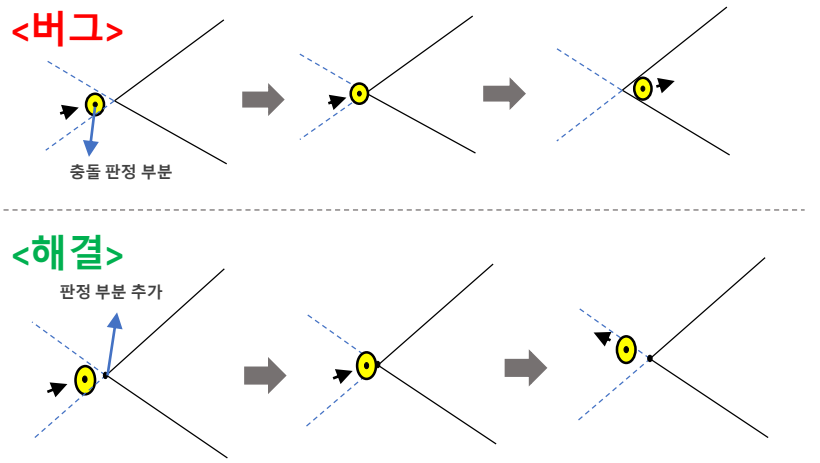
case 3: Projects

문제 발생 & 해결

1) <벽을 통과하다.>

- **이유** : 충돌 범위 사각지대.
  충돌 처리 시, 벽의 범위에 있는지 먼저 확인하는데,
  그 범위에 포함되지않아 충돌처리에 버그발생

- **해결** : 벽 사이, 사각지대에 **점**을 만들어
  그 점들 과도 충돌처리하기



2) <특정 각도의 벽에서 제대로 충돌 처리가 되지 않음.>

- **이유** : 충돌 처리 과정에서 atan(...) 함수를 사용한다.
  해당 함수는 결과값이 -π / 2 ~ π / 2 까지로 360º에 대한 결과를 도출하지 못함.

- **해결** : atan2(...)를 사용하여, -π ~ π 까지의 결과를 도출.

< atan(...) >

Return Value

Principal arc tangent of *x*, in the interval [-pi/2,+pi/2] radians.
One *radian* is equivalent to 180/PI *degrees*.

< atan2(...) >

Return Value

Principal arc tangent of *y*/*x*, in the interval [-pi,+pi] radians.
One *radian* is equivalent to 180/PI *degrees*.

[출처] www.cplusplus.com

case 3: Projects

Lv.2 - 복수의 원 (3학년)

+ **개요**
  // 다각형의 세상에서 핍박 받는 '원'을
  플레이하여 다각형들에게 복수하는 게임

+ **배경**
  // "게임 프로그래밍" 과목 프로젝트로,
  Unity엔진을 이용하여 개발,
  게임 개발에 필요한 기초적인 지식 습득.

+ **장르**
  // 2D 슈팅게임

+ **개발 인원**
  // 1인 개발

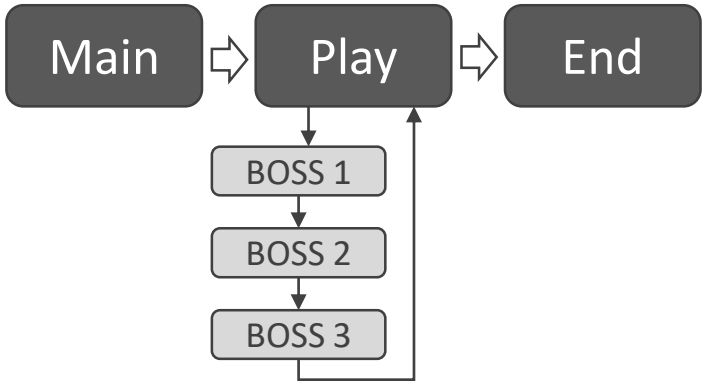
+ **개발언어 및 툴**
  // 언어 : C#
  // 개발 툴 : Unity
  // 플러그인 : DoTween

+ **개발 기간**
  // 2016.10 ~ 11

+ **링크**
  // **[소스코드]**
  - https://github.com/gangjung/TingTongTing

case 3: Projects

전체 로직(Scene)



Manager

+ 매 번 특정 GameObject를 찾아 특정 데이터를 얻어오는 작업은 부하가 많은 작업이며, 가독성도 매우 떨어짐.

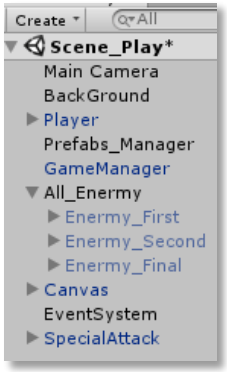
+ Manager를 만들어 작업 부하를 줄이고, 관리를 편하게 함.

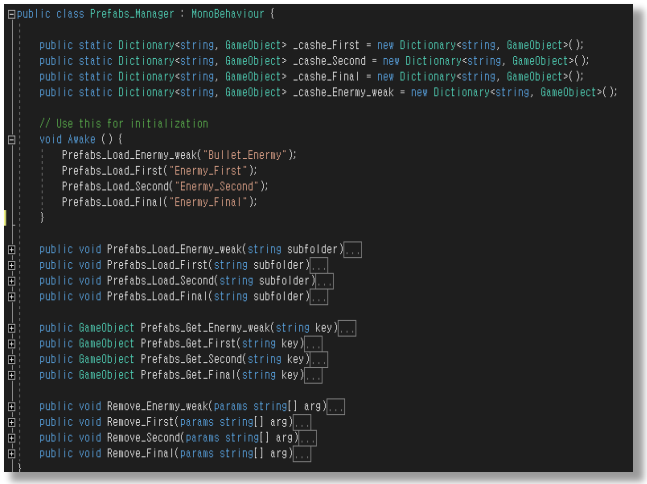
Game Manager

+ // 게임 전반적인 데이터나, 공통적으로 자주 사용되는 함수에 쉽게 접근함.

Prefabs Manager

+ // 게임 시작 시, 게임에서 사용될 Prefab 리소스들을 미리 Cashing하여 오브젝트 생성 속도를 향상시킴

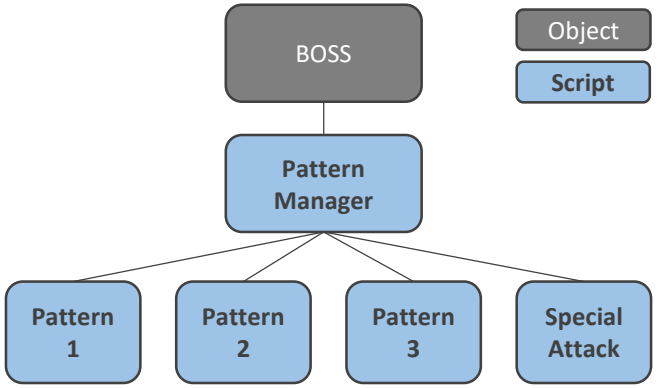




case 3: Projects

패턴 관리

+ 효율적인 관리를 위해, Pattern Manager를 이용하여 패턴 관리.
+ 가독성이 좋아지며, 각 Pattern에 대해 독자적으로 관리 가능.
+ 랜덤으로 Pattern이 동작하며, 4번째 마다 Special Attack 실행




```
void OnEnable()
{
    if (!runRoutin_Second)
    {
        runRoutin_Second = true;

        switch (state)
        {
            case PHASE.ONE:
                GetComponent<Pattern_Enermy_Second_1>().enabled = true;
                check_Random_Same = PHASE.ONE;
                break;

            case PHASE.TWO:
                GetComponent<Pattern_Enermy_Second_2>().enabled = true;
                check_Random_Same = PHASE.TWO;
                break;

            case PHASE.THREE:
                GetComponent<Pattern_Enermy_Second_3>().enabled = true;
                check_Random_Same = PHASE.THREE;
                break;

            case PHASE.SPECIAL:
                StartSpecialAttack();
                check_Random_Same = PHASE.SPECIAL;
                break;
        }

        count_Special_Attack++;

        if (count_Special_Attack == 3)
        {
            state = PHASE.SPECIAL;
            count_Special_Attack = 0;
        }
        else
        {
            do
            {
                state = (PHASE)Random.Range(0, 3);
            } while (state == check_Random_Same);
        }
    }
}
```

LookPlayer()

+ 삼각 함수를 이용, 플레이어가 있는 방향 값을 반환.
+ 대부분의 보스 패턴에서 사용.

```
Quaternion LookPlayer()
{
    Vector3 vectorToTarget = GameObject.Find("Player").transform.position - transform.position;

    // Mathf.Rad2Deg -> 라디안 to 각도.
    float angle = Mathf.Atan2(vectorToTarget.y, vectorToTarget.x) * Mathf.Rad2Deg;


    // AngleAxis는 해당 축을 기준으로 angle만큼 이동시키겠다는 함수이다.
    // angle + 90하는 이유는 발사하는 방향이 y축이기 때문이다.
    return Quaternion.AngleAxis(angle - 90, transform.forward);
}
```

Enum

+ 코드의 가독성을 향상시키기 위해 열거형 사용.

```
public enum STATE { WAIT, CREATE, ALIVE, DIE}
public enum SHIELD { NORMAL, DISTROYED }
public enum WEAPON { NORMAL, ROUNDBALL }
public enum PHASE { ONE, TWO, THREE, FOUR, SPECIAL}
public enum BULLET_SPHERE_STATE { ROLLING_LEFT,ROLLING_RIGHT, FIRE }
public enum PATTERN_B_F { PATTERN_1_R, PATTERN_1_L, PATTERN_2 }
```

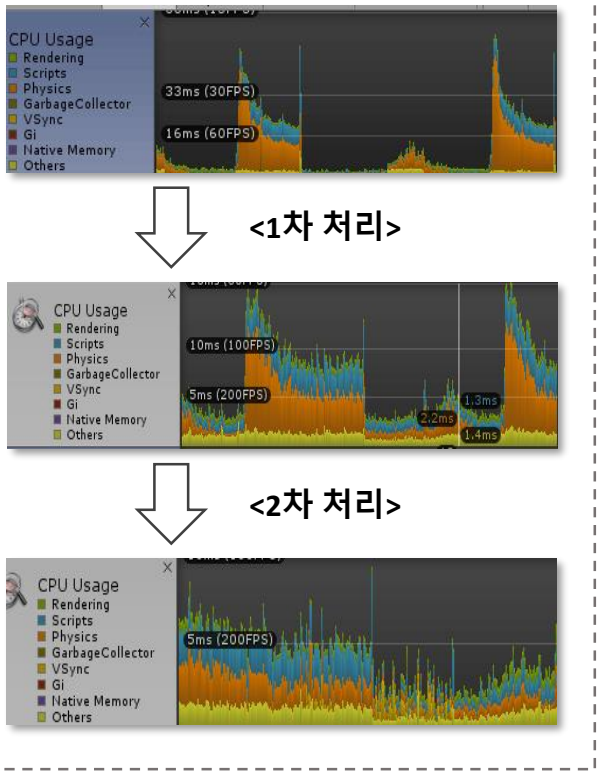
case 3: Projects

문제 발생 및 해결

1) <심각한 프레임 드랍 발생> – 1

- **이유** : **무분별한 Object 생성으로** 인해, 불필요한 작업 발생.
  무분별한 Instantiate(…) + Collider처리

- **해결** : Instantiate(…)는 부하가 심한 작업이기에 불필요한 작업이다.
  Object Pool 방식을 사용, **필요한 Object만 생성**하여
  Pool에서 위치 이동 및 Object 활성화 하여 사용.
  사용 후, Object 비활성화 하여 Pool로 **Push!**
  Instantiate(…) 및 불필요한 Collider 처리 제거.



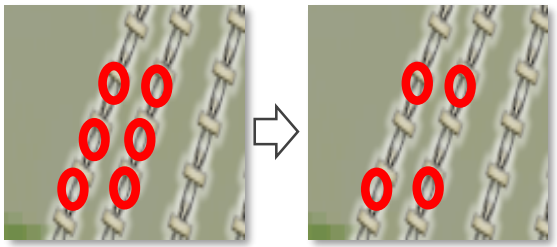
2) <심각한 프레임 드랍 발생> - 2

- **이유** : 무분별한 **Object 충돌**로 인해, 불필요한 작업 발생.

- **해결** : 패턴 특성 상, 총알의 이동속도가 빠르다.
  그 점을 이용해서, 모든 Object가 아닌,
  부분적으로 Collider를 적용해서 충돌처리 작업을 **줄임**.

플레이어는 '내부'를 모른다!

눈속임을 주자! "



case 3: Projects

페이지 번호

Lv.3 - 뻐끔뻐끔(4학년)

+ 수상
// 2017 공개SW 개발자대회 **-동상-** 수상

+ 개요
// 물고기 비서 및 어항 자동관리 시스템

+ 배경
// Android 및 서버와의 상호작용에 대한 지식 습득.
// 팀 프로젝트를 통해 협업에 대한 기초 지식 습득.

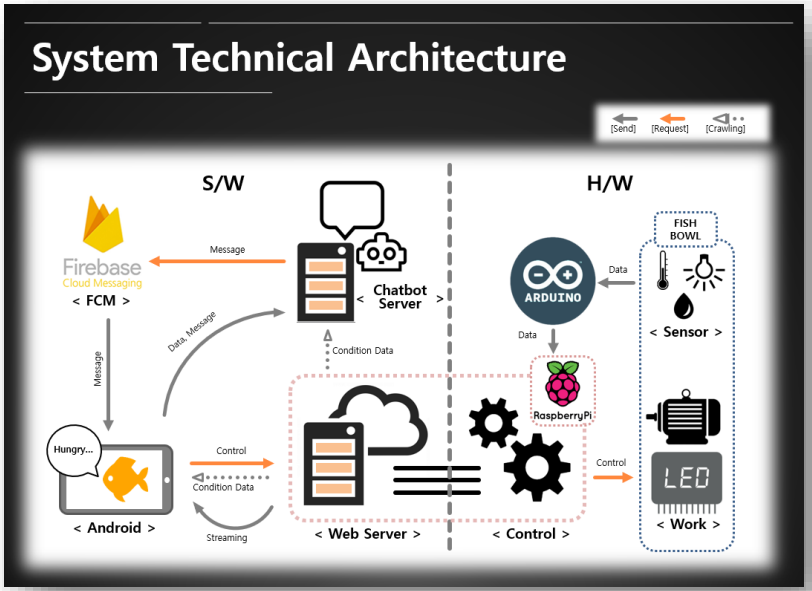
+ 장르
// IoT

+ 개발 인원 & 개발 기간
// 3인 개발
// 2017.05 ~ 07

+ 부분
// Android(Client) 설계 및 구현
// UI & Document 디자인

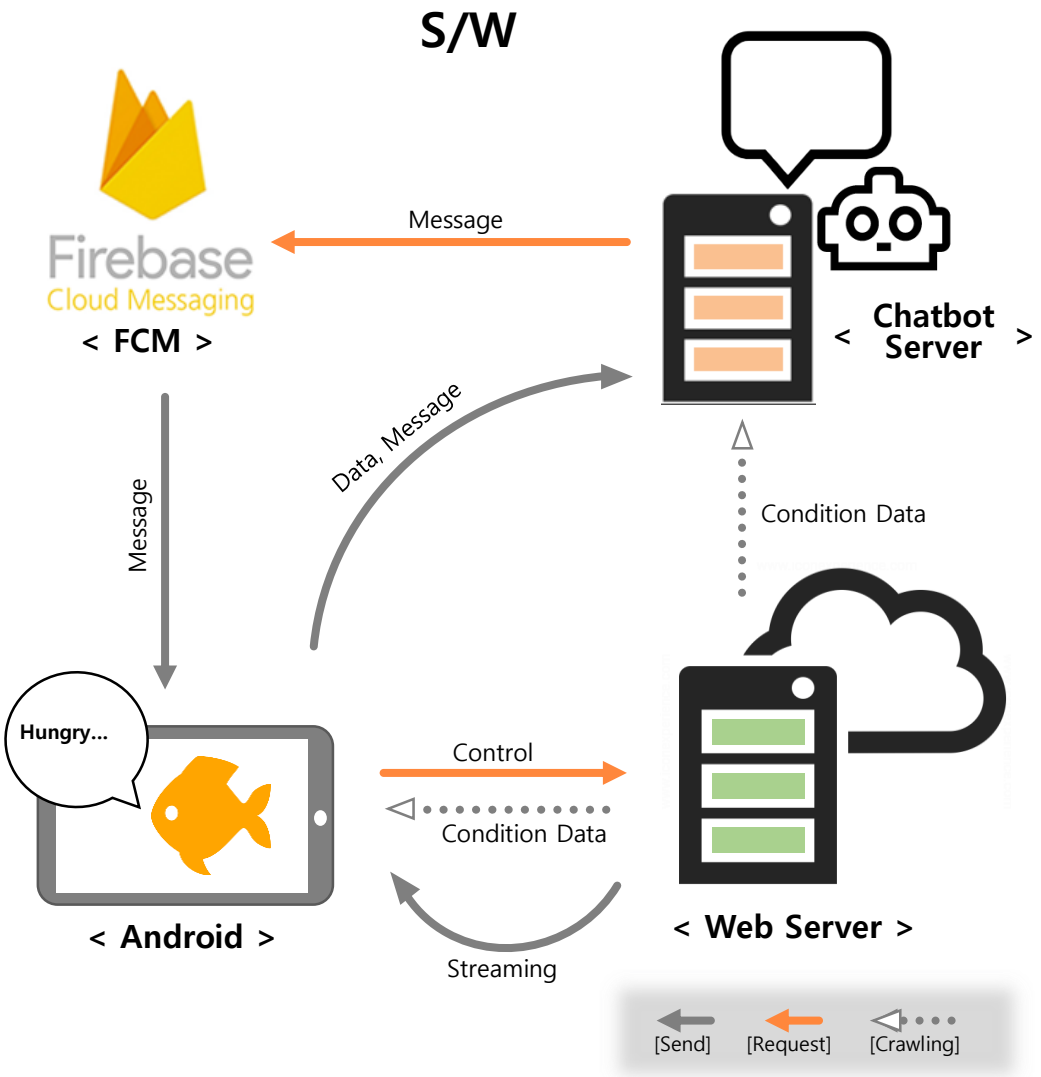
+ 개발언어 및 툴
// 언어 : Java
// 개발 툴 : Android Studio
// 라이브러리 : FCM, Google Calendar

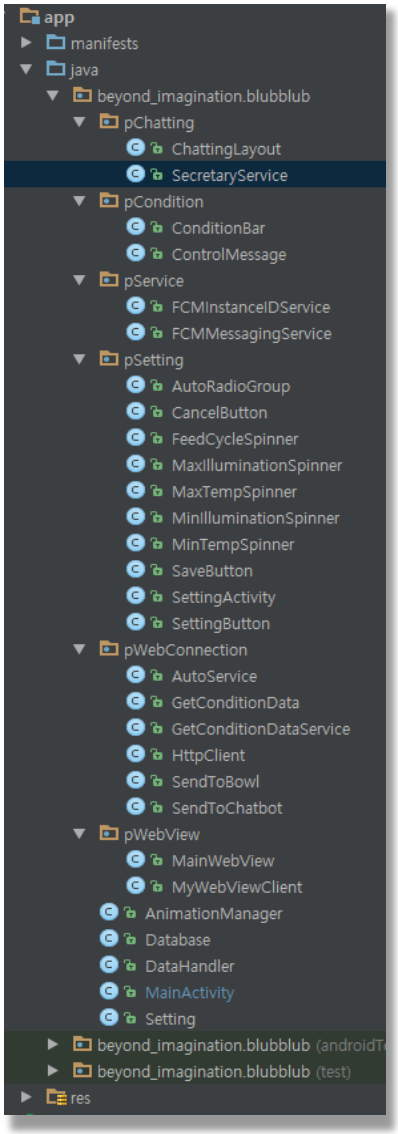
+ 링크
// **[소스코드]**
(전체) https://github.com/Beyond-Imagination/BlubBlub
(부분) https://github.com/gangjung/BlubBlub/tree/master/Android
// **[영상]** : https://www.youtube.com/watch?v=uU10i3ZTdqc



case 3: Projects

주요 로직



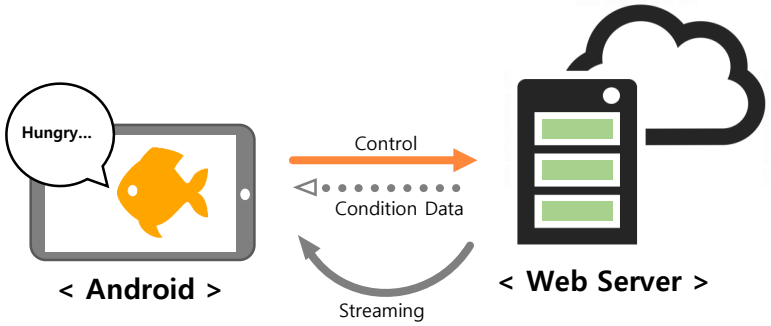


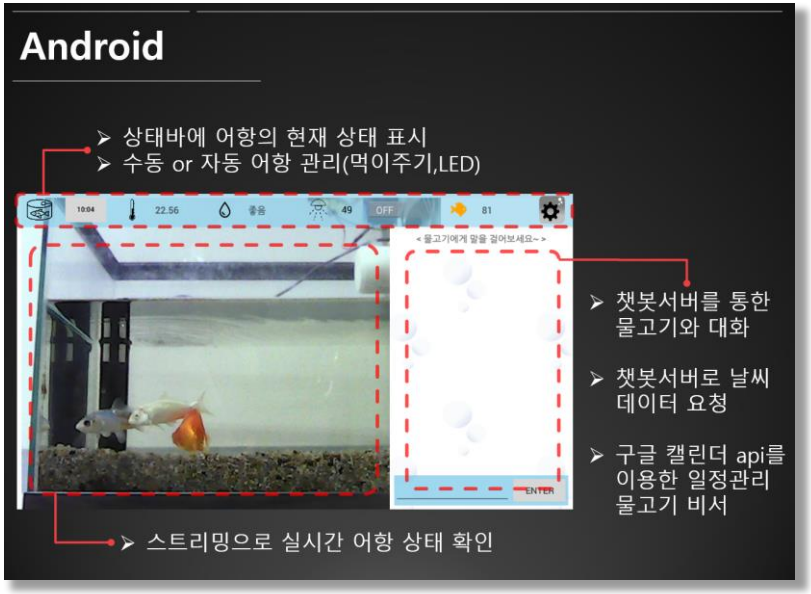
case 3: Projects

비동기 통신 – Web Server

+ 모바일 특성 상, 동기 방식으로 통신을 하기 어렵다.
  그래서 웹서버를 이용한 비동기 통신을 선택하였다.

+ 필요한 상황에만 명령 요청을 할 수 있도록,
  비동기 방식을 선택하여 불필요한 작업 제거.





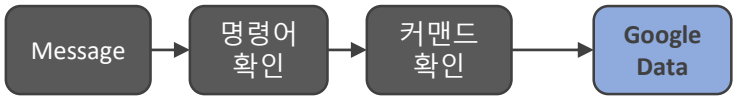
Google API(Google Calendar)

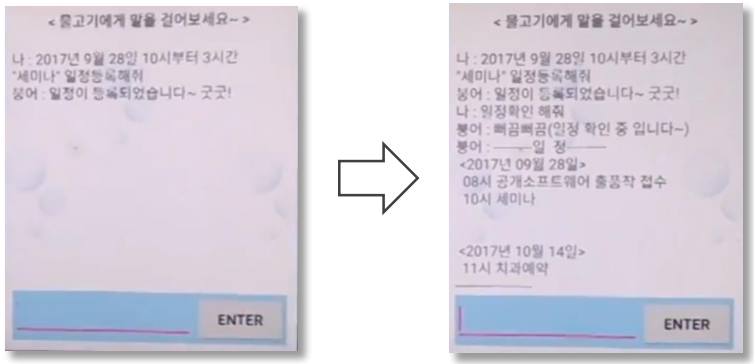
+ 사용자들이 대부분 가지고 있는 Google Account Data를 이용하여,
  보다 쉬운 Calendar 데이터에 접근 및 개발 가능.

+ (유저입장) Google API를 이용하면, Google Service가 이용가능한
  어느 곳이든 데이터 동기화가 가능하다.

+ 명령어를 설정하고, 그에 따른 Format을 확인하여,
  Calendar Data 송-수신

< 문자열 파싱을 통해 확인 >



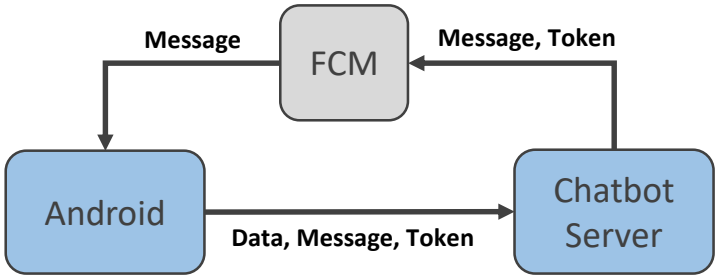


case 3: Projects

FCM

+ 서버에서 해당 기기의 IP값을 유지하기 힘듦.
  FCM을 이용하여 언제 어디서든 데이터 전송 및 수신.

+ Push 알림을 통해 어항의 특정 상태를 알려준다.



Http Post - Json

+ 데이터 송,수신에 효율적인 Post방식을 선택.

+ Json 형식의 데이터를 주고 받는다.

+ 원활한 통신을 위해, 회의를 거쳐 Data Key Set 설정.

```
Identity{
    type - token
    token - (토큰값)
    secret - (시크릿 값)
}

Control{
    token - (토큰값)
}
```

```
Setting{
    type - Setting
    feedcycle - 먹이 주기
    maxtemp - 최대 온도
    mintemp - 최소 온도
    maxillum - 최대 밝기
    minillum - 최소 밝기
}

Message{
    type - message, weather
    message - message
    token - (토큰값)
}
```

보안을 위한 토큰값 사용

+ 웹서버를 이용하여 하드웨어를 제어하기에,
  외부에서의 쉽게 접근 할 수 있다. 그렇기에,
  해당 기기마다 가지고 있는 고유 토큰 값을 이용하여,
  서버에서 사용자를 판별, 하드웨어를 제어한다.

```
/**
 * @brief
 * Get Url and connection with Url.
 * Use HttpURLConnection and just use output stream for control message
 * Write data that you want to send and execute .getResponseCode().
 * @param url
 */
public void getConnection(String url) {
    try {
        URL url_t = new URL(url);

        HttpURLConnection conn = (HttpURLConnection) url_t.openConnection();

        Log.d("SendToBowl", "getConnection execute");
        if (conn != null) {
            conn.setConnectTimeout(20000);
            conn.setRequestMethod("POST");
            conn.setDoOutput(true);

            Log.d("SendToBowl", "Data : " + data);
            DataOutputStream outputStream = new DataOutputStream(conn.getOutputStream());
            outputStream.writeBytes(data);

            int rescode = conn.getResponseCode();
            Log.d("SendToBowl", "responseCode" + rescode);

            outputStream.flush();
            outputStream.close();

            conn.disconnect();
        }
    } catch (ProtocolException e) {
        e.printStackTrace();
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

```
try {
    data.put("type", type);
    data.put("message", message);
    data.put("token", token);

    return data.toString();

} catch (JSONException e) {
    e.printStackTrace();
}
```

case 3: Projects

문제 발생 및 해결

1) <Backgound에서 얻은 데이터로 메인 UI 변경 불가능.>

- **이유** : Background(Sub Thread)에서 UI(Main Thread)를 제어할 수 없음.

- **해결** : 1) Message Handler를 이용하여 Sub Thread에서 Message를 전달, Main Thread 제어.
  2) AsyncTask를 이용하여 Background 작업 및 Main Thread 작업을 연동.

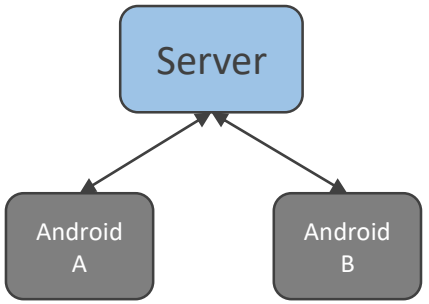
```
@Override
public void handleMessage(Message msg) {
    super.handleMessage(msg);

    bundle = msg.getData();

    Log.d("MainActivity", "Handler - " + msg.what);

    switch (msg.what) {
        case UPDATE_CONDITION:
            conditionBar.onConditionUpdate(bundle.getString("feedtime"), bundle.getS
            break;
```

```
* This work execute per every 10 second.
*/
public class GetConditionData extends AsyncTask<MainActivity, String, String> {

    MainActivity mainActivity;
    boolean isRunning = false;

    @Override
    protected void onPreExecute() {...}

    @Override
    protected void onPostExecute(String s) { Log.d("Http_result", s); }

    @Override
    protected void onProgressUpdate(String... values) {
        super.onProgressUpdate(values);

        mainActivity.onConditionUpdate(values[0], values[1], values[2], values[3]);
    }

    @Override
    protected String doInBackground(MainActivity... params) {
        mainActivity = params[0];
```

2) <여러 기기 이용 시, 어항 옵션 값이 통일되지 않음>

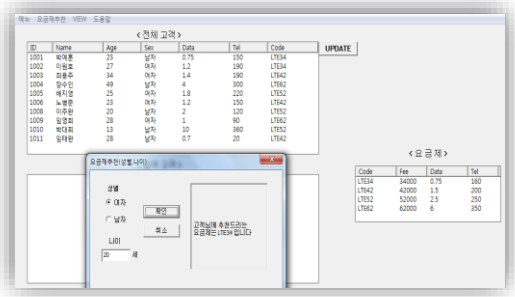
- **이유** : 각 기기별로 내부 DB로 값을 저장하기 때문.
  다수 접근을 생각하지 않고, 단일 접근만 생각.

- **해결** : Data를 주고 받을 수 있는 Chatbot Server에 어항 옵션 값 저장.
  어플리케이션 실행 시, Server에서 어항 옵션 값을 받음.
  옵션 값 변경 시, 변경된 값 Server에 저장.



case 3: Projects - Others

Lv.1 – 통신사 고객관리 시스템(2학년)

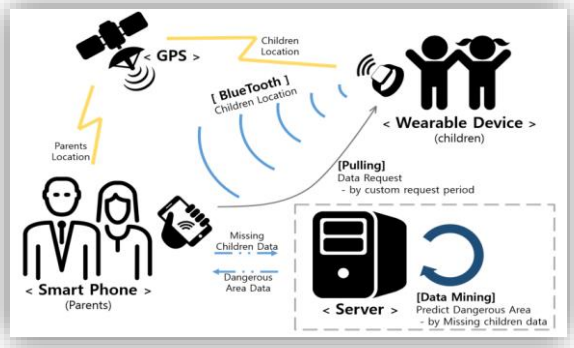


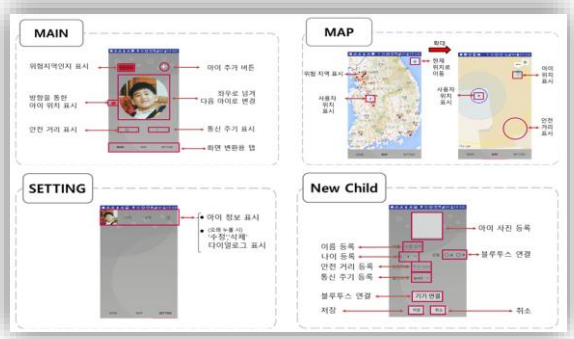
+ // 개요
  - Win API와 MySQL을 연동한 통신사 고객관리 시스템

+ // 개발 인원
  - 3인 개발

+ // 개발언어 및 툴
  - 언어 : C
  - 라이브러리 : Win API, MySQL

Lv.3 – 미드미아(4학년)





+ // 개요
  - 미아방지 프로그램 및 웨어러블 디바이스

+ // 장르
  - IoT

+ // 개발 인원
  - 3인 개발

+ // 부분
  - Android(Client) 설계 및 구현
  - UI & Document 디자인

+ // 개발언어 및 툴
  - 언어 : Java
  - 라이브러리 : Google Map API
  - 툴 : Android Studio

+ // 개발기간
  - 2017.07 ~ 09

+ // 수상
  - 한이음 공모전 -입선-

+ // 영상
  - https://www.youtube.com/watch?v=OD5ZukzdYc

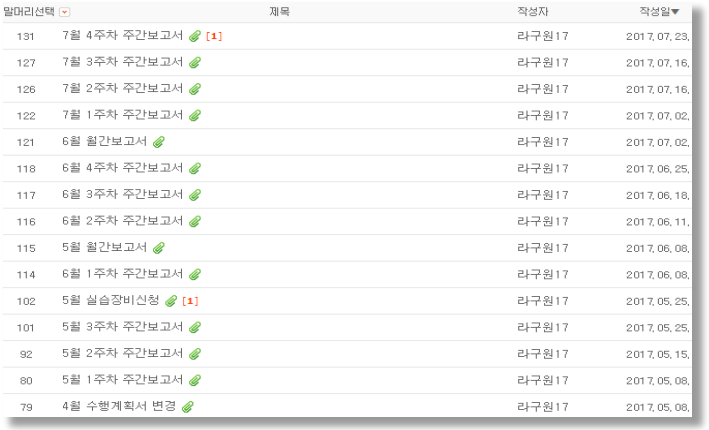
case 4: Co-Work & Management

정기 회의 및 개발보고서 작성

+ 매 주 또는 달 마다, 모든 팀원이 모여 오프라인 회의를 진행함.
+ 서로의 진행 상황 및 결과를 확인하여, 차 후 진행 방향을 수립함.

(가장 중요한 부분!!!)

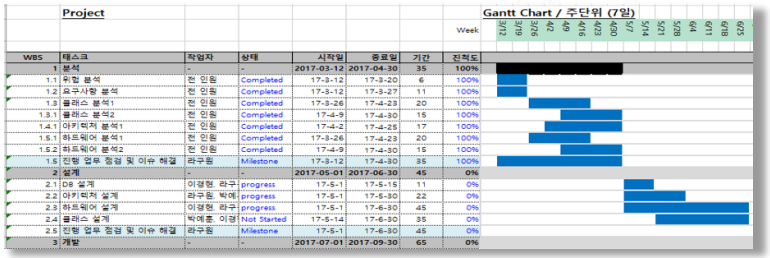
+ 회의를 통해, 서로 맡은 부분에 대한 지식을 얻을 수 있고, 결과가 좋지 못한 팀원에게 좋은 자극을 줄 수 있음.
+ [대표 링크] : http://cafe.naver.com/strartfrom



말머리선택	제목	작성자	작성일
131	7월 4주차 주간보고서 [1]	라구원17	2017.07.23.
127	7월 3주차 주간보고서	라구원17	2017.07.16.
126	7월 2주차 주간보고서	라구원17	2017.07.16.
122	7월 1주차 주간보고서	라구원17	2017.07.02.
121	6월 월간보고서	라구원17	2017.07.02.
118	6월 4주차 주간보고서	라구원17	2017.06.25.
117	6월 3주차 주간보고서	라구원17	2017.06.18.
116	6월 2주차 주간보고서	라구원17	2017.06.11.
115	5월 월간보고서	라구원17	2017.06.08.
114	6월 1주차 주간보고서	라구원17	2017.06.08.
102	5월 실습장비신청 [1]	라구원17	2017.05.25.
101	5월 3주차 주간보고서	라구원17	2017.05.25.
92	5월 2주차 주간보고서	라구원17	2017.05.15.
80	5월 1주차 주간보고서	라구원17	2017.05.08.
79	4월 수행계획서 변경	라구원17	2017.05.08.

WBS(Work Breakdown Structure)

+ 원활한 프로젝트의 진행을 위해, 프로젝트 하부 단위의 모든 작업들을 체계적으로 분류하여 관리함.
+ 전체 개발 시간 및 기간을 확인하기위해 사용함.

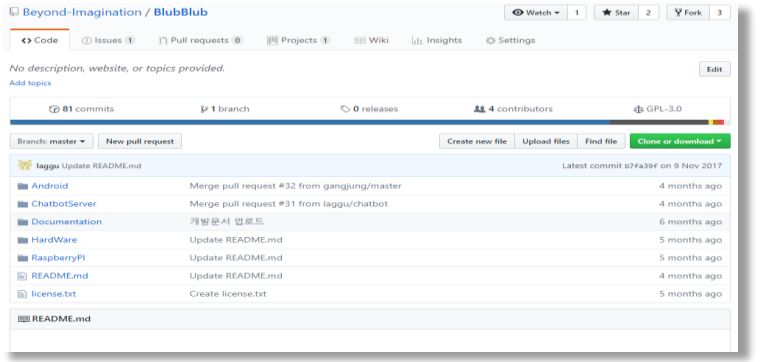


Project — Gantt Chart / 주단위 (7일)

WBS	태스크	작업자	상태	시작일	종료일	기간	진척도
1	분석	-	-	2017-03-12	2017-04-30	35	100%
1.1	위협 분석	전 인원	Completed	17-3-12	17-3-20	6	100%
1.2	요구사항 분석	전 인원	Completed	17-3-12	17-3-27	11	100%
1.3	클래스 분석1	전 인원	Completed	17-3-26	17-4-23	20	100%
1.3.1	클래스 분석2	전 인원	Completed	17-4-9	17-4-30	15	100%
1.4.1	아키텍처 분석1	전 인원	Completed	17-4-2	17-4-25	17	100%
1.5.1	하드웨어 분석1	전 인원	Completed	17-3-26	17-4-23	20	100%
1.5.2	하드웨어 분석2	전 인원	Completed	17-4-9	17-4-30	15	100%
1.5	진행 업무 점검 및 이슈 해결	라구원	Milestone	17-3-12	17-4-30	35	100%
2	설계	-	-	2017-05-01	2017-06-30	45	0%
2.1	DB 설계	이경현 라구	progress	17-5-1	17-5-15	11	0%
2.2	아키텍처 설계	라구원 박에	progress	17-5-1	17-5-30	22	0%
2.3	하드웨어 설계	이경현 라구	progress	17-5-1	17-6-30	45	0%
2.4	클래스 설계	박에홍 이경	Not Started	17-5-14	17-6-30	35	0%
2.5	진행 업무 점검 및 이슈 해결	라구원	Milestone	17-5-1	17-6-30	45	0%
3	개발	-	-	2017-07-01	2017-09-30	65	0%

GitHub

+ GitHub를 통해 서로의 작업물을 공유하여, 프로젝트의 전체적인 소스 및 버전 관리를 함
+ Open Source로 활용하여 사람들과 소스 공유.
+ [대표 링크] : https://github.com/Beyond-Imagination/BlubBlub



Beyond-Imagination / BlubBlub

Watch 1 | Star 2 | Fork 3

Code | Issues 1 | Pull requests 0 | Projects 1 | Wiki | Insights | Settings

No description, website, or topics provided. Edit

Add topics

81 commits | 1 branch | 0 releases | 4 contributors | GPL-3.0

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

laggu Update README.md — Latest commit b7fa39f on 9 Nov 2017

Android	Merge pull request #32 from gangjung/master	4 months ago
ChatbotServer	Merge pull request #31 from laggu/chatbot	4 months ago
Documentation	개발문서 업로드	6 months ago
HardWare	Update README.md	5 months ago
RaspberryPI	Update README.md	5 months ago
README.md	Update README.md	4 months ago
license.txt	Create license.txt	5 months ago

README.md

Thank you

Park YeHun