<본 보고서는 ver2 로 새로 갱신될 예정입니다.>

FINAL RESULT REPORT _ver1

<AR TCG Card Battle>

Project Team

Team 3

Date

2020-06-18

Team Information
컴퓨터공학과 201611300 조승현
산업디자인학과 201512755 현인수
기계공학과 201310805 장혁준

Table of Contents

1 개요

본 프로젝트는 건국대학교 졸업프로젝트1의 'AR CARD BATTLE'팀(3팀)의 최종 결과보고서이다. 팀은 조승현, 장혁준, 현인수로 구성되어 있다.

1.1 계획서

프로젝트는 2020년 4월에 시작되어 6월 3주차까지 진행되었으며 주제선정, 설계, 구현, 디버깅의 과정으로 진행되었다. 본래 6월 초까지 마무리되어야 했던 일정에 따라 촉박하게 계획을 세웠으나 최종발표일이 변경됨에 따라 개발계획이 수정되었다.

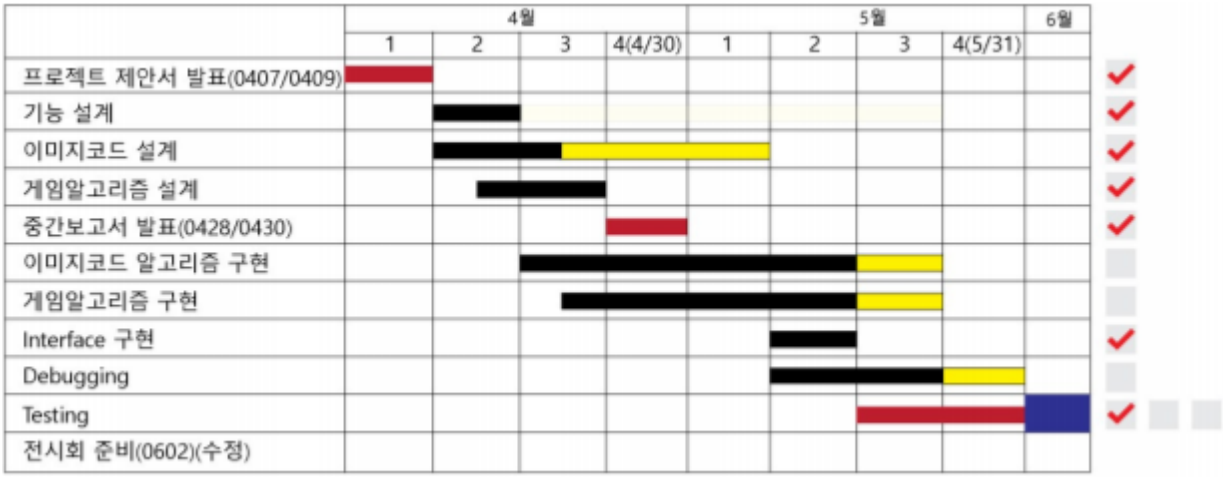
다음은 Iteration 1 중간보고서발표 0408의 계획서이다.

- Schedule_1

	4월				5월				6월
	1	2	3	4(4/30)	1	2	3	4(5/31)	
프로젝트 제안서 발표(0407/0409)									
기능 설계									
이미지코드 설계									
게임알고리즘 설계									
중간보고서 발표(0428/0430)									
이미지코드 알고리즘 구현									
게임알고리즘 구현									
Interface 구현									
Debugging									
최종보고서 발표(0526/0528)									
전시회 준비(0602)									

학기초에 작성한 계획서이기 때문에 개발과 디버깅에 상당한 시간을 약속하였다. 그러나 아래의 0430의 계획서를 보면 설계를 비롯하여 개발 및 디버깅 일정까지 연기된 것을 확인할 수 있다.

- Schedule_2



Iteration 2였던 0604일 구현의 70%가 끝났지만 모듈을 개발하고 연결하는 과정에서 상당한 시간을 소모하였다.

1.2 기획

1.2.1 의도

기존의 컴퓨터 카드게임이 가지고 있는 시각적 효과라는 이점과 실재 카드게임이 가지고 있는 수집하는 재미라는 이점을 모두 살린 AR Card Game 개발을 목적으로 한다.

<그림1. 기존의 컴퓨터 카드게임 및 실재 카드게임>

1.2.2 Game Logic

1.2.2.1 요약

- 게임 로직에 따라 게임이 진행되는지 확인한다.
- 플레이어 턴 변경 확인
- 누적공격력을 받을 경우 생명력이 깎이는지 확인
- 공격카드를 냈을 때 누적 공격력이 카드의 공격력만큼 증가하는지 확인
- 몬스터카드 / 보조카드 / 마법카드 / 저주카드가 속성에 맞게 작동하는지 확인

1.2.2.2 상세

1.2.2.2.1 Card(실물)

1) 몬스터카드

몬스터 공격력은 0부터 100까지의 정수값을 가진다. 일부 몬스터 카드는 특수한 효과를 지닌다.

2) 보조카드

사용시 데미지를 줄여주거나 무시할 수 있는 효과를 지닌 카드이다.

3) 마법카드

자신의 라이프를 일정 수치만큼 회복하거나 저주를 풀 수 있는 카드이다.

4) 저주카드

특정한 대상을 지정하여 저주를 걸 수 있다. 저주는 다른 저주카드로 덮어쓸 수 있으며, 덮어지기 전까지 지속된다.

1.2.2.2.2 Rules

1) 게임 세팅

테이블 위에는 게임 화면을 표시하기 위한 테블릿 PC가 있다. 그 옆에 카드 뭉치와 카드를 낼 수 있는 공간이 있다. 플레이어가 내는 카드는 이 위에 누적하여 쌓이게 된다.

2) 게임 시작

게임을 시작하기 위하여 플레이어 인원수를 테블릿에서 실행 중인 게임 소프트웨어에 입력한다. 소프트웨어 상 인원수만큼 플레이어 1,2,3,4로 등록되며 각각 라이프를 200씩 받게 된다. 이후 플레이어들은 카드 뭉치에서 5장씩 카드를 나누어 갖는다. 플레이어들은 가위바위보 등을 통해 순서를 정한다.

3) 게임 진행

플레이어는 본인의 차례에 카드를 한 장 내거나 버릴 수 있다. 카드를 내는 경우 카메라를 통해 카드를 인식하여 게임이 자동으로 진행된다. 카드를 버릴 경우 모니터 상의 카드버리기 버튼을 누르면 된다. 첫 번째 플레이어가 카드를 내거나 버리면 게임은 자동으로 시작된다.

라이프는 200을 초과할 수 있지만 0이 되면 플레이어는 패배하고 게임에서 제외된다. 자신의 차례에 한 장 내거나 버린 뒤에 카드 뭉치에서 한 장을 가져온다. 자신의 차례가 끝날 때, 플레이어는 누적 데미지 수치가 0이 아닌 경우 해당 수치만큼 라이프가 깎이게 되고, 누적데미지는 0이 된다. 이후 다음 플레이어에게 차례가 넘어간다.

▶ 플레이어가 낸 카드가 몬스터카드일 경우

해당 몬스터 카드가 효과를 지닌 카드일 경우 카드의 효과를 적용한 후 데미지를 계산한다.

효과가 없는 몬스터 카드일 경우 바로 데미지를 계산한다. 데미지 계산은 다음과 같은 방식으로 이루어진다.

내 카드의 공격력이 같거나 더 낮은 경우 누적 데미지를 입은 후, 누적데미지를 내 몬스터카드의 공격력으로 설정한다.

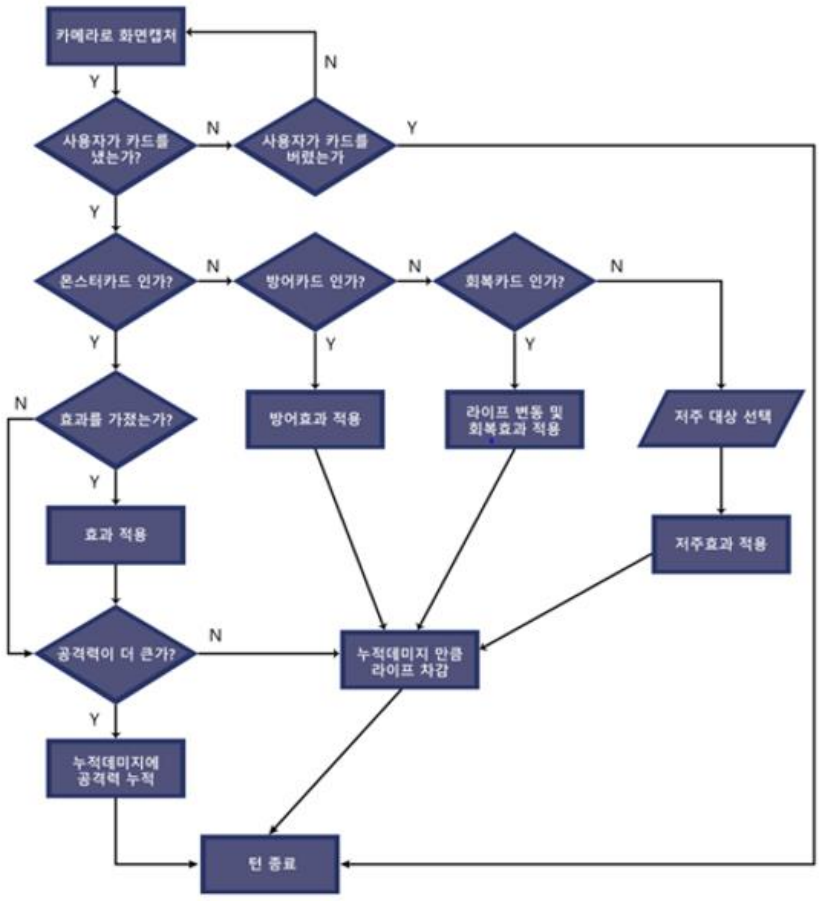
내 카드의 공격력이 더 높은 경우 누적 데미지에 내 몬스터카드의 공격력을 더한다.

이전 플레이어가 낸 카드가 몬스터 카드가 아닐 경우 누적데미지를 내 몬스터카드의 공격력으로 설정한다.

▶ 플레이어가 낸 카드가 보조카드일 경우
마지막 낸 카드가 몬스터 카드일 경우 보조카드의 수치만큼 누적데미지를 줄이거나, 누적데미지를 무시하거나, 상대방에게 누적데미지를 전달할 수 있다. 이후 누적데미지만큼 라이프가 깎인다.
▶ 플레이어가 낸 카드가 마법카드일 경우
마법카드의 각 속성에 따라 효과가 발동된다.

▶ 플레이어가 낸 카드가 저주카드일 경우
특정 대상을 선택하여 저주를 걸 수 있다. 이후 누적데미지가 0이 아닌 경우 수치만큼 라이프가 깎인다.

1.2.3 Flow Chart



2. SRS(Software Requirements Specification)

2.1 Functional Requirement

2.1.1 카드를 인식했을 경우 Card Recognition 모듈을 통해 코드를 해독한다.
2.1.2.1 카메라를 통해 캡쳐된 영상을 분석하여 해당 이미지에 코드데이터가 존재하는지 확인한다.

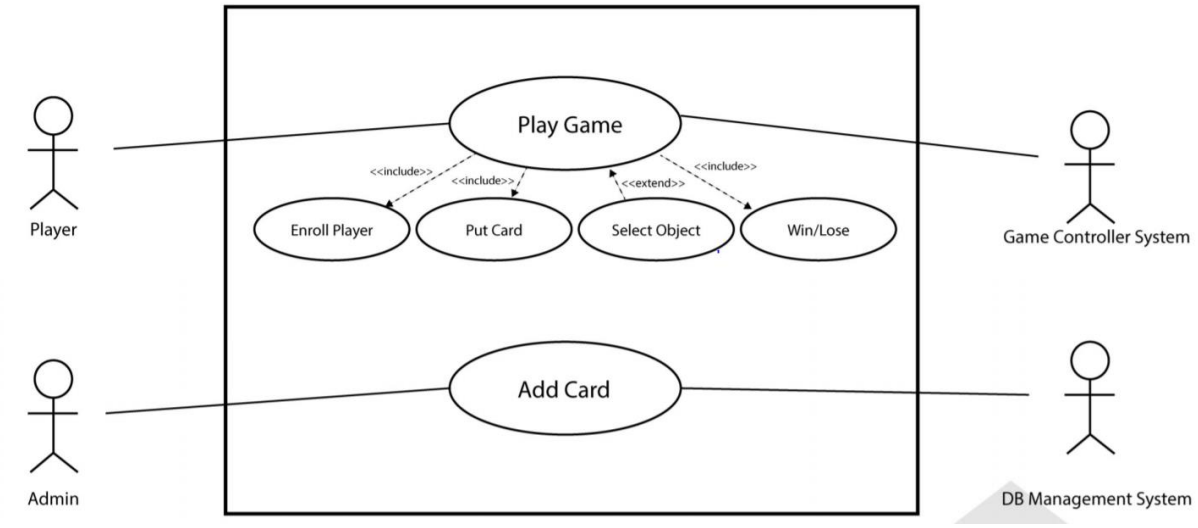
2.1.2.2 코드를 찾았을 경우 코드를 방향과 각도에 상관없이 데이터를 읽을 수 있게 보정처리 한다. 보정처리한 코드를 해독 모듈로 넘겨준다.
2.1.3 플레이어 라이프에 변동이 있거나, 차례가 바뀔 때 해당 정보를 Display Controller에 업데이트 해 준다.
2.1.4 읽어온 카드 데이터를 가지고 알맞은 행동을 취하여 게임을 진행시킨다.
2.1.5 플레이어가 몬스터 카드를 냈을 경우 해당 몬스터의 그래픽을 화면에 띄워준다.
2.1.6 코드를 해독해 해당 카드의 종류가 어떤 종류인지 (0 : 몬스터카드, 1 : 보조카드, 2 : 마법카드, 3 : 저주카드), 몇 번 카드인지 (1,2,....) 정수 데이터를 얻어낸다. 이후 데이터 파서로 해당 정수 값이 전달된다.
2.1.7 해독된 정수 데이터를 가지고 DB에서 검색한 후 카드 데이터를 읽어온다.
2.1.8 복호화한 코드 데이터를 가지고 DB에서 카드의 정보를 읽어올 수 있다.
2.1.9 카드를 신규 제작하였을 때, 해당 카드 정보를 DB상에 추가할 수 있다.

2.2 Non-Functional Requirement

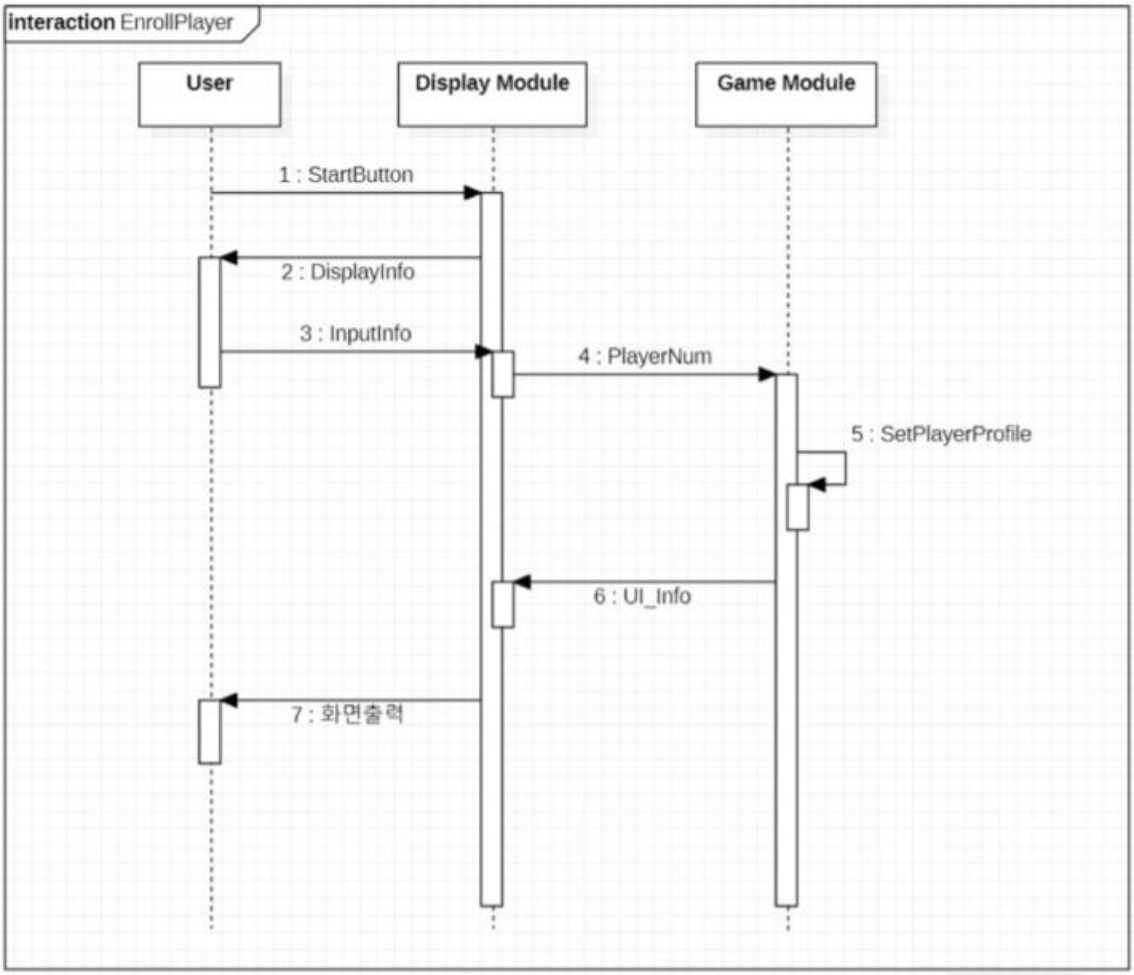
2.2.1 사용자가 카드를 테이블에 내려놓은 순간부터 카메라는 해당 카드를 3초 이내에 인식 완료하여야 한다.
2.2.2 이 게임을 한 번도 해보지 않은 플레이어도 화면을 보고 1초 이내에 화면을 보고 1초 이내에 수치를 읽을 수 있도록 라이프와 데미지의 UI요소들을 가시성과 가독성이 좋게 배치한다.

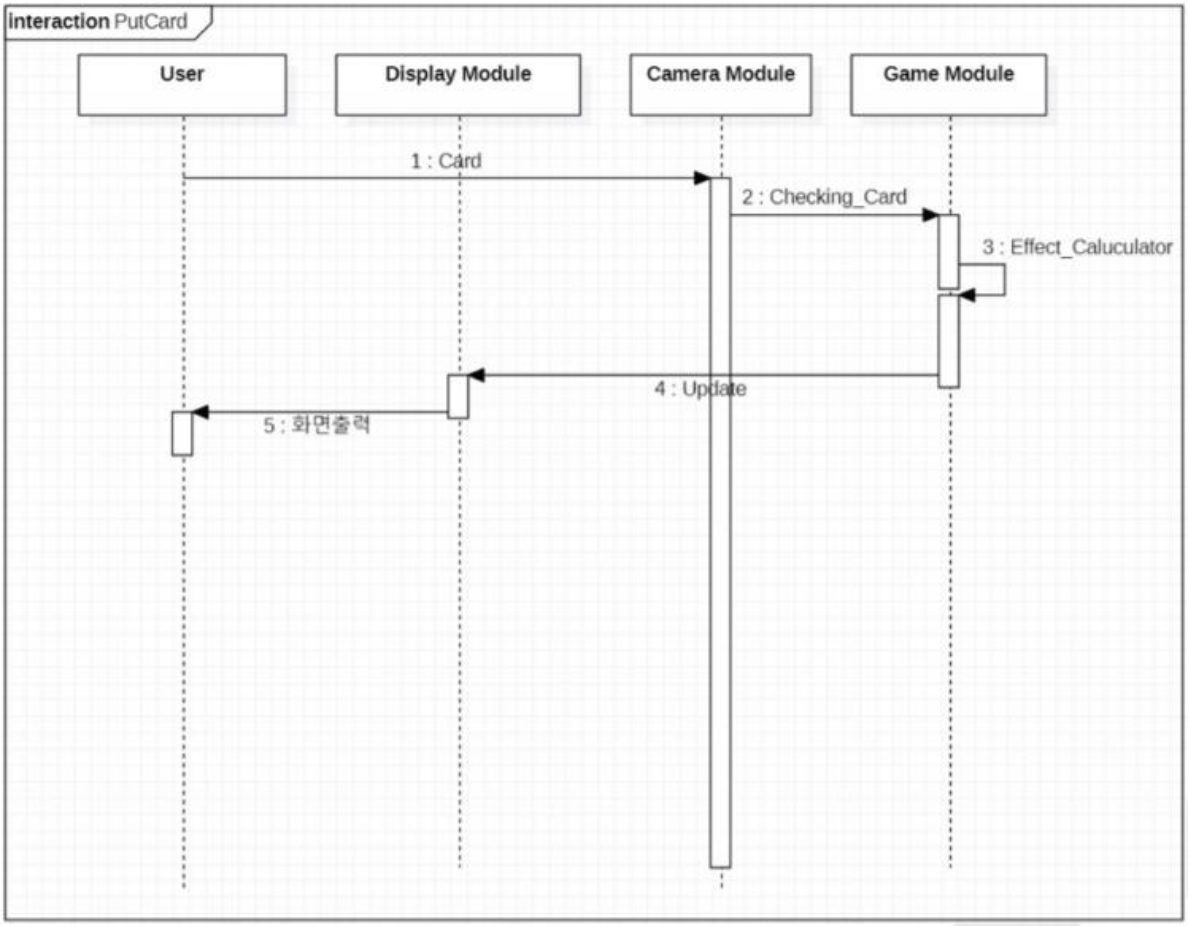
2.3 High Level Design

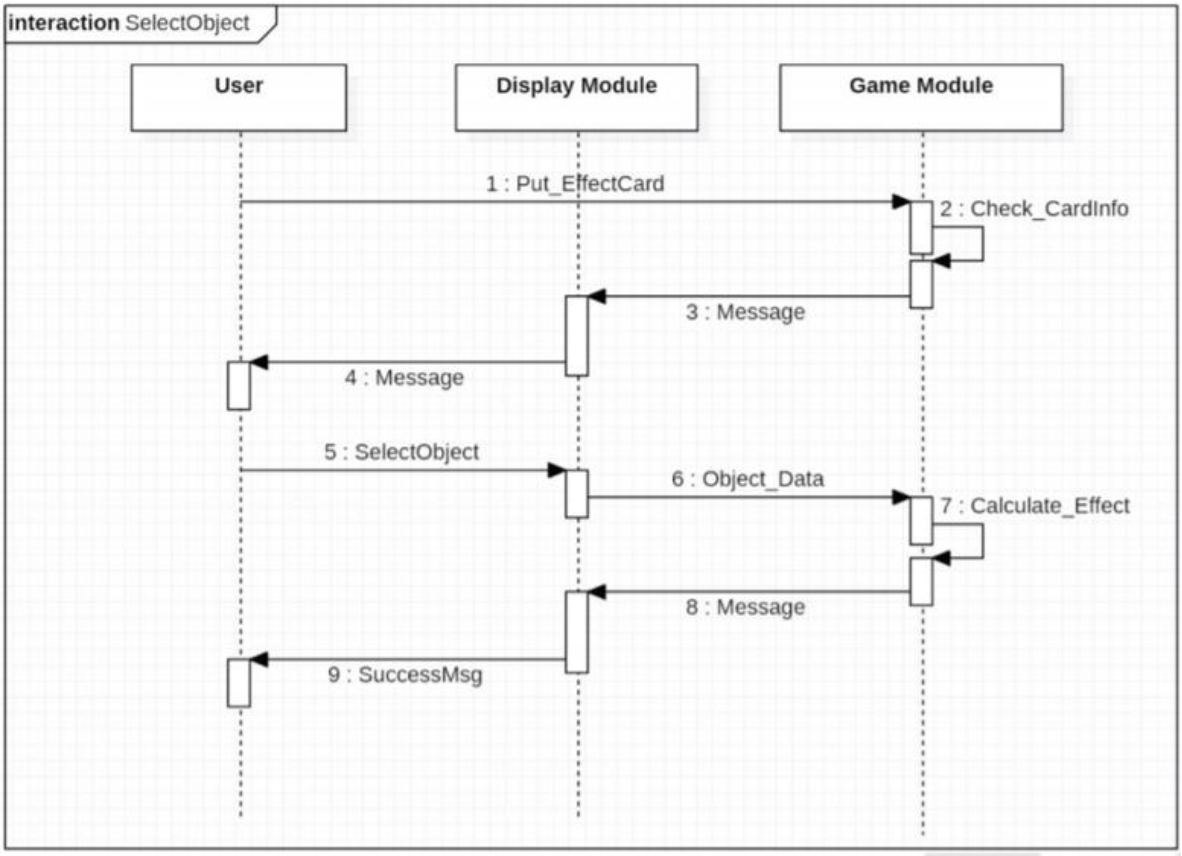
2.3.1 Use Case Diagram

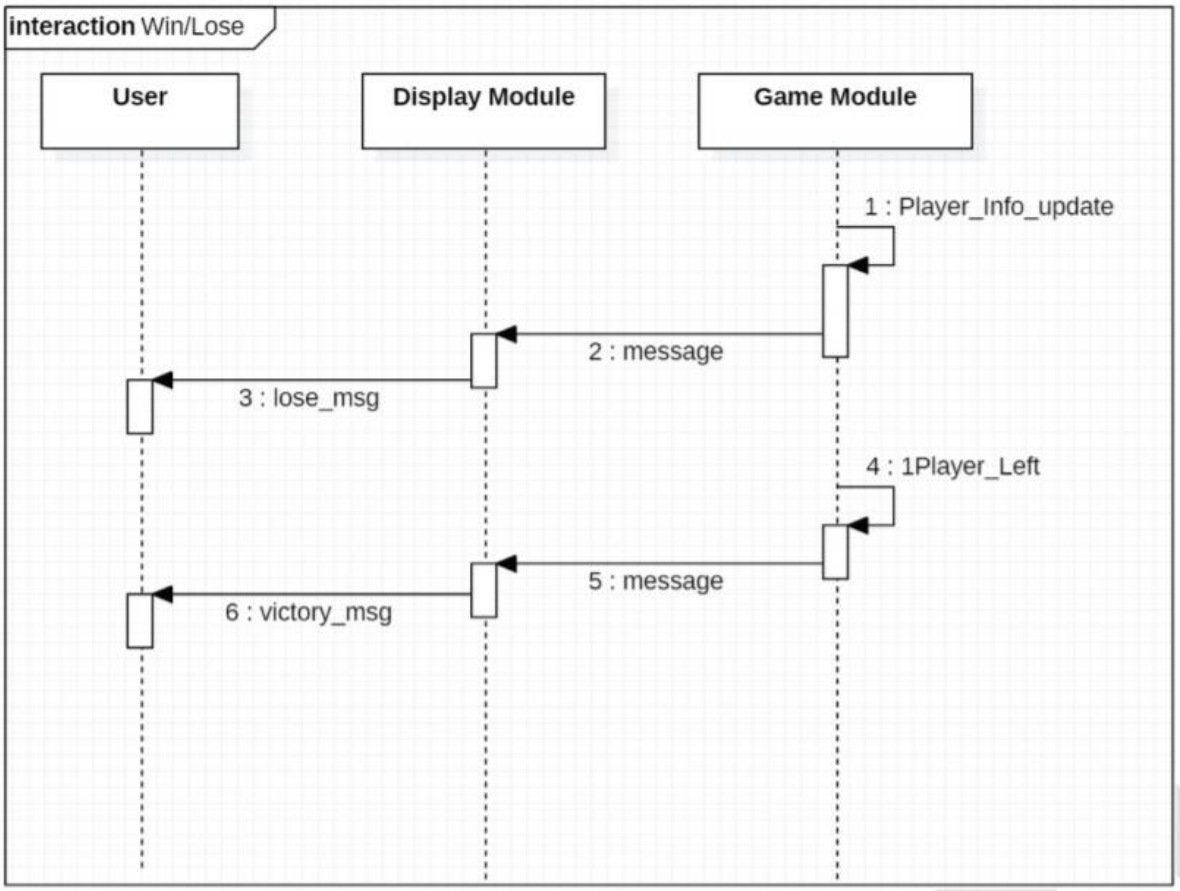


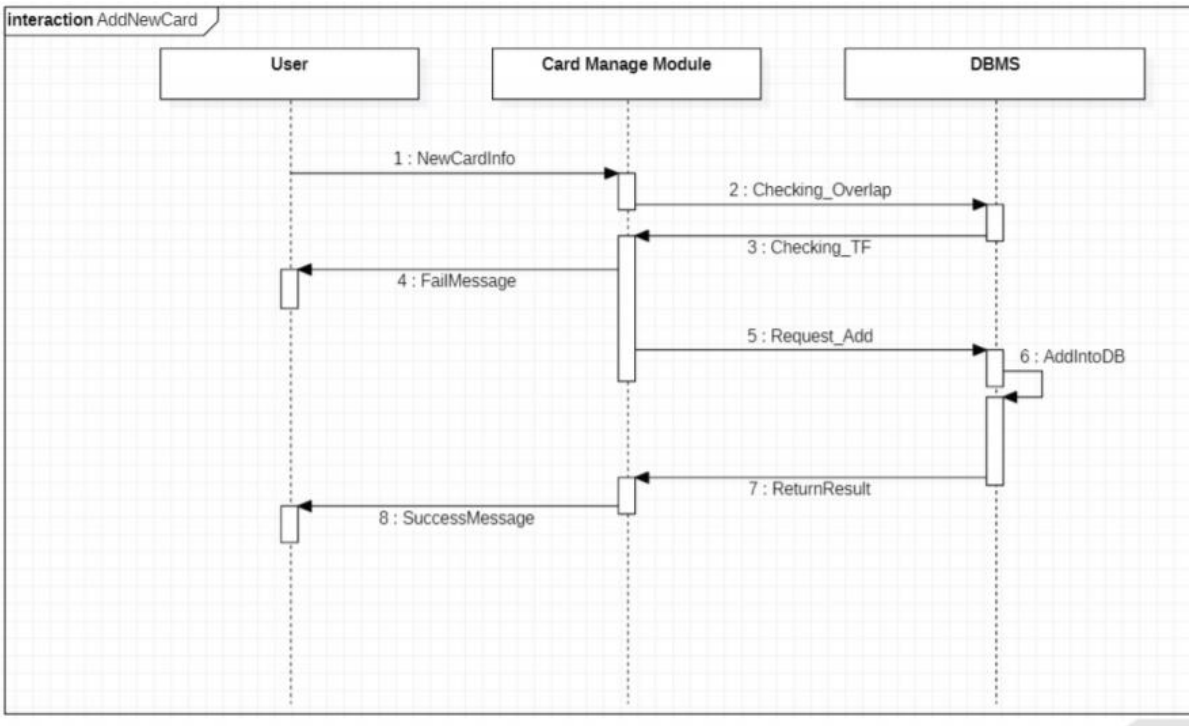
2.3.2 Sequence Diagram



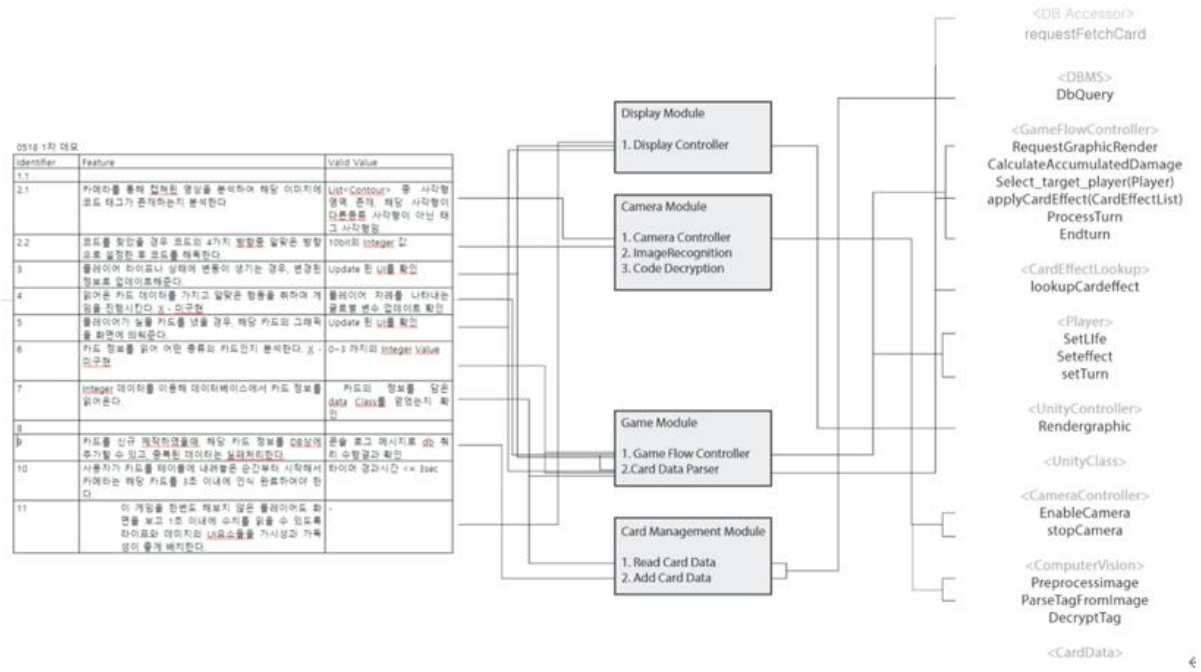

interaction PutCard
User
Display Module
Camera Module
Game Module
1 : Card
2 : Checking_Card
3 : Effect_Caluculator
4 : Update
5 : 화면출력


interaction SelectObject
User
Display Module
Game Module
1 : Put_EffectCard
2 : Check_CardInfo
3 : Message
4 : Message
5 : SelectObject
6 : Object_Data
7 : Calculate_Effect
8 : Message
9 : SuccessMsg


interaction Win/Lose
User
Display Module
Game Module
1 : Player_Info_update
2 : message
3 : lose_msg
4 : 1Player_Left
5 : message
6 : victory_msg

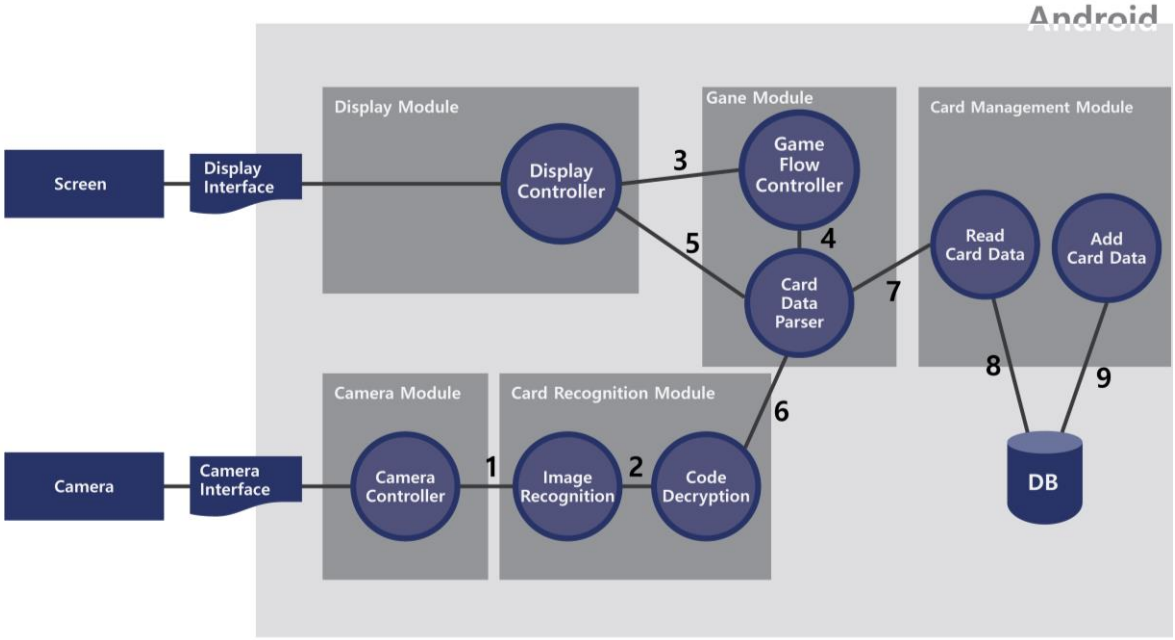

interaction AddNewCard
User
Card Manage Module
DBMS
1 : NewCardInfo
2 : Checking_Overlap
3 : Checking_TF
4 : FailMessage
5 : Request_Add
6 : AddIntoDB
7 : ReturnResult
8 : SuccessMessage

2.4 Traceability



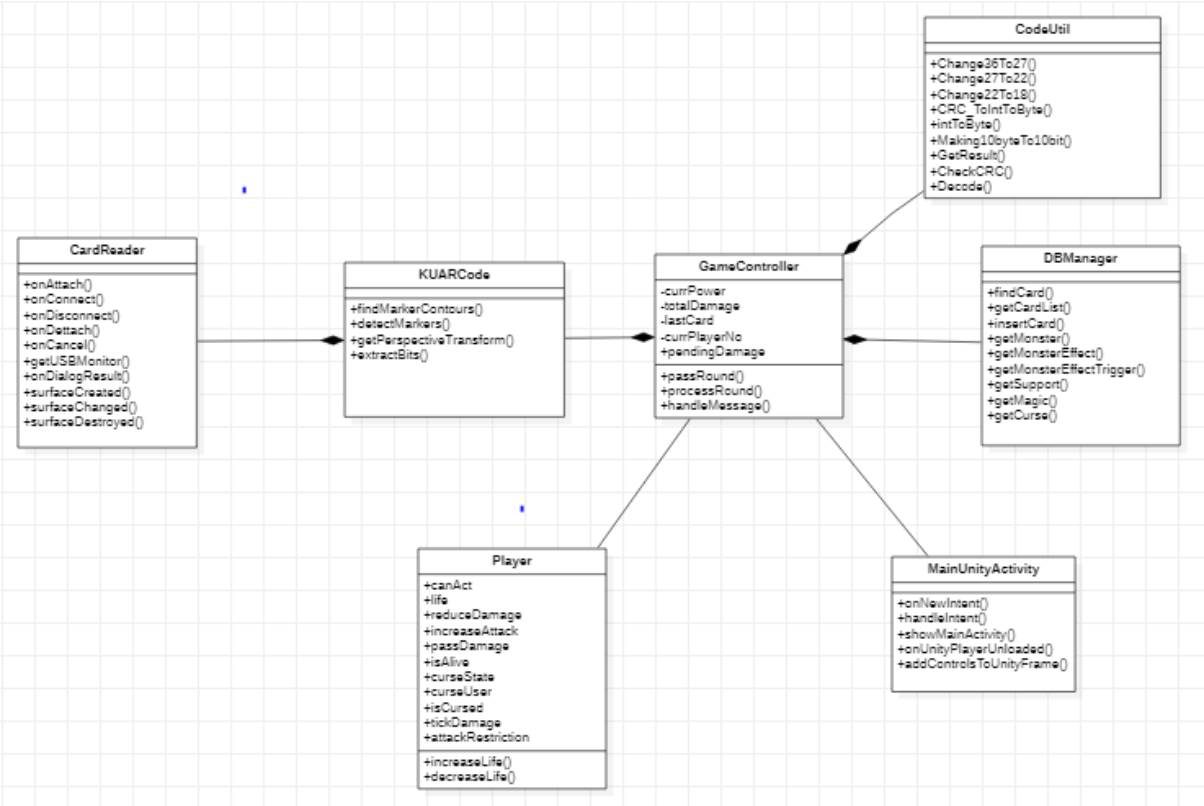
3 SDS(Software Design Specification)

3.1 Architecture Diagram

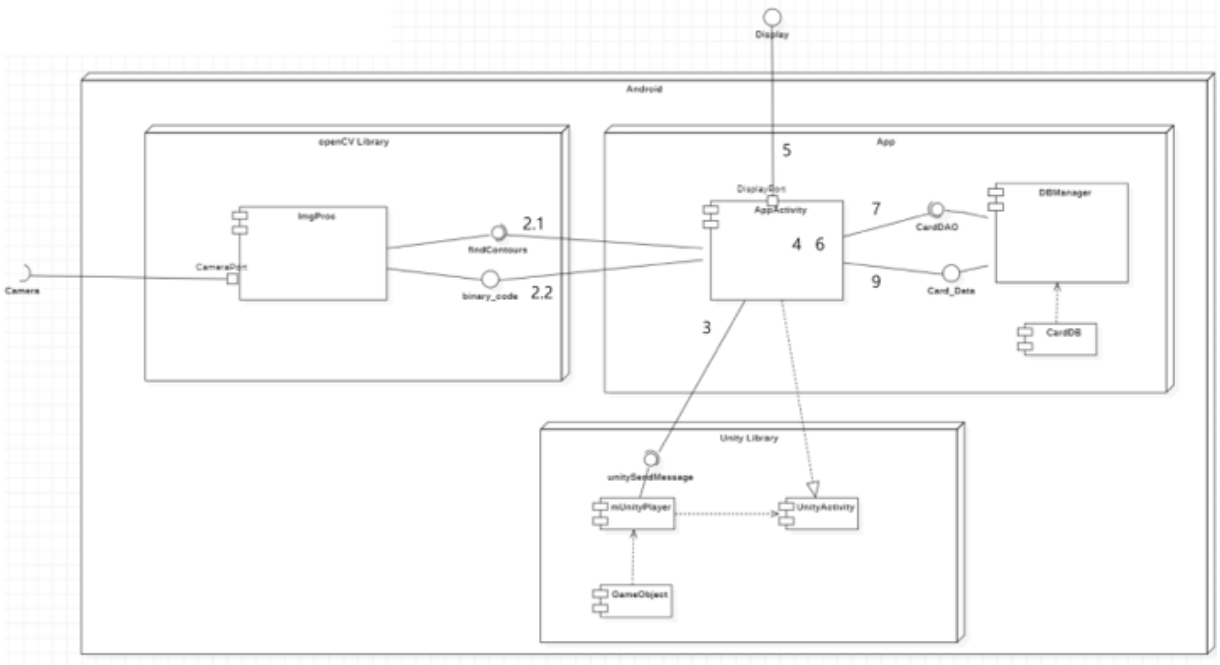


3.2 Low Level Design

3.2.1 Class diagram



3.2.2 Component Diagram



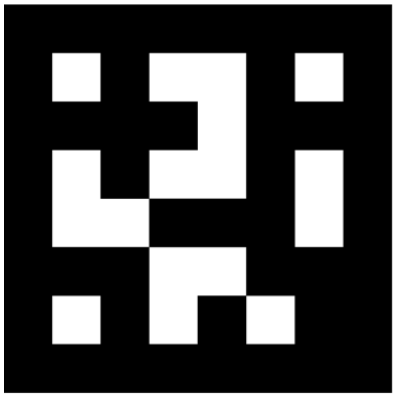
4 Develop Process

4.1 2D CODE 개발

4.1.1 Encoding - 카드 정보를 담기 위한 8X8 2D Code를 만든다.

- Code영역 검출을 위해 가장자리의 28bit를 할당한다.
- 코드의 상하좌우를 식별하기 위하여 왼쪽 위, 오른쪽 위, 왼쪽 아래 3bit씩 총 9bit를 할당한다.
- 총 10비트를 데이터정보를 위해 할당하여 약 1000장의 카드 데이터를 입력할 수 있다.
- 데이터 암호화를 위해 CRC코드 8bit와 Hamming Code 4bit를 추가한다.
- f(x) 함수를 적용하여 27bit의 코드를 만든다.

4.1.2 Decoding - 카드에 2D Code를 삽입하여 카메라로 인식할 수 있도록 한다.



<KUARCODE>

<게임 카드>

4.2 Computer Vision 처리과정

4.2.1) Android Studio 에서 OpenCV library 사용하기 위한 import 과정

우선, import module로 Opencv android 버전을 받아주고 dependencies에서 모듈을 추가하여 Opencv를 사용할 수 있도록 한다. 주로 많이 사용되는 라이브러리는 Imgproc과 Core이다. Imgproc은 ImageProcess를 하기 위한 함수 라이브러리이고, Core는 자료형들을 다루고 있다.

이미지 포맷은 bytebuffer, bitmap, mat이 주로 사용되는데 본 프로젝트에서 사용한 OpenCV라이브러리는 그 중에 Mat을 다룬다. 영상을 처리하기 위해 OpenCV를 import하고 Activity에 OnCameraFrame(inputFrame) 이라는 override된 함수가 있지만 실제 개발 과정에서는 callback안에 CameraDialog interface의 onFrame이라는 함수를 매 프레임마다 불러 사용한다. onFrame의 파라미터에는 frame이 들어가는데 이는 bytebuffer로 처리되어 있고 본 프로젝트에서는 Mat의 자료형으로 바꾸어 영상처리를 진행한다. 처음 프레임을 받아올 때는 RGBA의 데이터로 받아오지만 물체를 인식하기 위해서는 frame을 바이너리화해야 하는데 이를 위해 회색조처리를 우선적으로 해야 한다.

4.2.2) 영상 보정

영상을 보정하기 위해 우선적으로 inputFrame을 받아야 한다. 본래 라이브러리를 이용한다면 Mat m_mat = inputFrame.rgba() 이지만 본 프로젝트에서는 bytebuffer의 inputFrame을 bitmap으로 바꾸고 이를 다시 Mat으로 바꾸어 사용한다.

4.2.2.1) 회색조 전환

회색조로 전환하기 위한 함수는 Imgproc.*cvtColor*(mat1, mat3, Imgproc.*COLOR_BGRA2GRAY*)
이다. 1번 파라미터는 source, 2번 파라미터는 destination matrix, 3번 파라미터는 전환하기 위한 코드값으로 변환하고자 하는 이미지의 형태이다.

4.2.2.2) Gaussian blur

GrayScale된 이미지의 noise를 제거하기 위해서 blur과정이 필요하다.
Imgproc.*GaussianBlur*(mat3, mat2, new Size(3, 3), 2,2)
파라미터는 순서대로 src, dst, blur하고자하는 주변픽셀, blur강도이다.

4.2.2.3) delate, erode 함수

이 함수들은 Canny라는 외곽선 검출 보정의 다음단계로 사용된다. delate는 검출된 부분을 과장해서 더 잘 인식되도록 바꿔주고 erode는 과장된 부분의 노이즈된 부분을 평균화해서 평탄하게 해주는 작업이다. 이를 통해 외곽선(contour) 검출에 있어 견고한 Matrix를 얻을 수 있다.
해당과정의 코드는 다음과 같다.

```
Size kss = new Size(3,3);
mask = Imgproc.getStructuringElement((Imgproc.MORPH_CROSS),kss);
Imgproc.dilate(mat2,mat3,mask,m,3);
Imgproc.erode(mat3,mat1,mask,m,3);
```

여기서 특이한 부분은 getStructuringElement 함수이다. blur와 마찬가지로 주변 픽셀을 mask로 사용한 MORPH_Cross는 말그대로 십자가형으로 마스크를 만든다. 파라미터는 순서대로 src, dst, mask, 중심점으로 자료형은 Point 이며, Point(-1,-1)을 기본으로 한다.

4.2.2.4) binary img 전환 (threshold) or 필요에 따라 Canny 함수를 사용

Threshold 함수를 이용하여 검정(0)과 흰색(1)으로 만들어 2차원코드 인식률을 높인다.
해당과정의 코드는 다음과 같다.
Imgproc.*threshold*(mat1,mat3,127,255,Imgproc.*THRESH_BINARY*);
파라미터는 순서대로 src, dst, threshold value1, threshold value2, Mode이다.
Threshold value1과 Threshold value2 사이의 gray값을 갖는 픽셀을 모두value에 맞게 0과 1로 바꾸고 이 과정에서 THRESH_BINARY 모드의 경우 검정에 가까운 픽셀은 검정으로, 흰색에 가까운 픽셀은 흰색으로 만들어준다. 이후 임의의 선분을 찾아 위치를 저장한다.

4.2.3) 컨투어 생성

그 선분은 다음과 같은 자료형으로 저장된다.

```
List<MatOfPoint> contours = new ArrayList<MatOfPoint>();

Mat hierarchy = new Mat();

Imgproc.findContours(mat3, contours, hierarchy, Imgproc.RETR_TREE, Imgproc.CHAIN_APPROX_SIMPLE);
```

contour는 Point들의 집합이며, 하나의 contour가 아닌 여러 개의 contour가 화면에서 저장된다. 따라서, 지정되는 최종 자료형은 List<MatOfPoint>가 된다. 이후 수많은 선분들 중 2차원코드를 선별해야 한다.

4.2.3.1) 컨투어 필터링 – 본 프로젝트의 findContour.

Contour는 MatofPoint자료형의 array로 저장되어 있는데 이를 for문으로 하나씩 가져와 선별한다. 본 프로젝트는 선별을 위한 조건으로 4가지의 처리를 한다.

- 1. Contour 사이즈가 일정 수준보다 작으면 제거한다. 이 과정에서 2가지 단계를 거친다.
- 1.1) 화면의 크기와 contour의 길이 차이가 일정 수준이상 날 경우 contour를 제거한다.
- 1.2) 전체 선분의 길이를 가져와 일정 값보다 작으면 contour를 제거하기 위해 contourArea()를 사용한다.
- 2. contour가 사각형인지 여부에 따라 제거한다.
- 3. 근사화를 통해 contour가 convex한지 여부에 따라 제거한다.
- 4. 화면 외곽부분에서 검출되는 contour는 본 프로젝트가 바라는 contour가 아닐 확률이 높기 때문에 일정부분 화면과 가까워지면 삭제한다.

4.2.4) 선별된 contour 중에서 2 차원코드를 검출해야 한다. - refineMatrix( )

4.2.4.1) 받아온 점 4 개에 대하여 외적하여 점을 시계방향 혹은 반시계방향으로 회전하며 맞게 정렬해야 한다.

해당 과정의 코드는 아래와 같다.

```
public Point GetVector( List<Point> a, int x, int y){ //x-y

double tempx = a.get(x).x - a.get(y).x;

double tempy = a.get(x).y - a.get(y).y;

Point newpoint= new Point(0,0);
```

```
newpoint.x = tempx;
newpoint.y = tempy;
return newpoint;
}
```

각 contour는 아래의 코드를 거쳐 정제된다.

```
List<Point> p_point = new ArrayList<>();
for (int k = 0; k < p_mat_of_point.size(); k++) {
//받아온 모든 contour 에 대하여
p_point = p_mat_of_point.get(k).toList();
Point v1 = GetVector(p_point, 1, 0);//내 다음 점과 내 위치를 외적,
Point v2 = GetVector(p_point, 2, 0);//내 대각 점과 내 위치를 외적.
double o = (v1.x * v2.y) - (v1.y * v2.x);
if (o < 0.0) {//swap
Point temp = new Point(0, 0);
temp.x = p_point.get(1).x;
temp.y = p_point.get(1).y;
p_point.get(1).x = p_point.get(3).x;
p_point.get(1).y = p_point.get(3).y;
p_point.get(3).x = temp.x;
p_point.get(3).y = temp.y;
}
}
```

4.2.4.2) Perspective 인 상태를 frontview 로 변환해야 한다.

getPerspectiveTransform(), warpPerspective()를 사용하여 contour를 원하는 사이즈에 맞게 변환한다. getPerspectiveTransform()은 변환행렬을 생성한 후 warpPerspective()로 변환한다.
이후 형태가 보간이 되어 회색 부분이 존재할 수 있어 Threshold를 한 번 더 한다.
만약 Canny처리가 된 frame이라면 2차원코드 내부의 값을 읽을 수 없기 때문에 refineMatrix에서는 threshold된 값을 받아야 한다. 본 프로젝트는 400 x 400의 frontview로 생성한다.

```
MatOfPoint2f src = new MatOfPoint2f(
new Point(p_point.get(0).x,p_point.get(0).y),
new Point(p_point.get(1).x,p_point.get(1).y),
new Point(p_point.get(2).x,p_point.get(2).y),
```

```
new Point(p_point.get(3).x,p_point.get(3).y)
);
MatOfPoint2f dst = new MatOfPoint2f(
new Point(0, 0),
new Point(400-1 , 0),
new Point(400-1 , 400-1 ),
new Point(0, 400-1 )
);
distMat = Imgproc.getPerspectiveTransform(src, dst); // 찾은 점을 dst 점으로 핀것의 행렬 = distMat
temp1 = mat3.clone();
Imgproc.warpPerspective(mat3, temp1, distMat, new Size(400, 400));//source, dest , m, size(최대 width, height)
Imgproc.threshold(temp1,temp2,127,255,Imgproc.THRESH_BINARY);
```

4.2.4.3) frontview를 자신의 코드의 사이즈에 맞게 분할하여 정제하는 작업.

8x8의 2차원코드를 400x400의 frontview로 받아왔기 때문에 이를 분할하기 위해 subMat(Rect(x,y))의 형태로 자르고, 이 Rect 안의 값의 절반이상이 byte array에 1일 경우 1로, 0일 경우 0으로 설정하여 2차원코드를 저장한다. 본 프로젝트에서는 bitMatrix로 저장한다.

픽셀의 값에 접근하는 것은 bitMatrix.get(y,x)[0]을 사용한다. 만약 여기서 rgba의 r 값에 접근하고자 한다면. bitMatrix.get(y,x)[3]으로 하면 된다. 값의 저장은 bitMatrix.put(y,x,data)로 저장한다.

4.2.4.4) 정제된 matrix 를 90 도씩 회전하면서 2 차원코드의 상하좌우를 찾아내는 작업

해당과정의 코드는 아래와 같다.

```
public Mat rotateMatCW(Mat src, double deg ){
Mat returnMatrix = new Mat();
if (deg == 270 ){
// Rotate clockwise 270 degrees
Core.transpose(src, returnMatrix);
Core.flip(returnMatrix, returnMatrix, 0);
```

```
}
else if (deg == 180 ){
// Rotate clockwise 180 degrees
Core.flip(src, returnMatrix, -1);
}
else if (deg == 90 ){
// Rotate clockwise 90 degrees
Core.transpose(src, returnMatrix);
Core.flip(returnMatrix, returnMatrix, 1);
}

return returnMatrix;
}
```

90도씩 돌리다가 값이 일치하면 리턴한다.

```
Mat rotation[] = new Mat[4];
rotation[0] = bitMatrix;

if (bitMatrix.get(0, 0)[0] == 1 && bitMatrix.get(0, 5)[0] == 1 && bitMatrix.get(5, 0)[0] == 1) {
RealCodeMatrix = rotation[0];
return RealCodeMatrix;
}
else{
for (int p = 1; p < 4; p++) {
// rotate(rotation[p - 1], rotation[p], Core.ROTATE_90_CLOCKWISE);
double l=90;
rotation[p] = rotateMatCW(rotation[p-1],l);
l+=90;
if (rotation[p].get(0, 0)[0] == 1 && rotation[p].get(0, 5)[0] == 1 && rotation[p].get(5, 0)[0] == 1) {
RealCodeMatrix = rotation[p];
return RealCodeMatrix;
}
```

```
}
}
```

4.2.5) 회전하여 코드로 인식된 경우 byte[] Arr 를 저장하고 이를 Decode 하는 작업.

Decoding과정은 Encoding의 역순으로 진행한다. 본 프로젝트에서는 byte array를 사용했지만 2차원코드의 이해를 돕기 위해 해당 문서에서는 bit로 설명한다.

4.2.5.1) 읽어진 RealCodeMatrix는 36bit코드이다. 현재 검정이 0, 흰색이 1로 저장되어 있지만 본 프로젝트에서는 검정을 1로 사용한 2차원코드를 사용하기 때문에 이를 반전시켜준다.

4.2.5.2) 인식점을 비롯한 테두리를 제외하여 27bit코드로 만들어준다.

4.2.5.3) 해당 과정 이후 f(x)의 역함수를 적용해서 코드를 복호화하여 22bit로 만들어준다.

4.2.5.4) hamming code 4bit를 확인하여 잘못 전달된 정보는 받지 않고, 성공 시에 18bit로 만들어준다. 본 프로젝트에서는 hamming code로 '1011'를 사용했다.

4.2.5.5) CRC code를 다시 XOR 해서 값이 맞아 떨어질 경우 전송에 성공하고, 이에 따라 10 bit의 데이터 코드를 리턴한다.

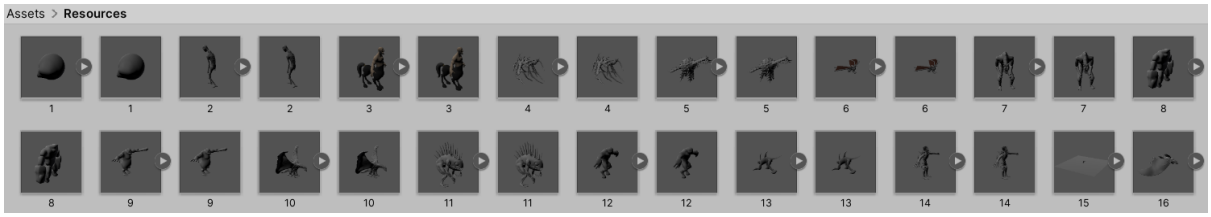
4.2.5.6) 데이터코드를 (Integer)Card_ID로 바꾸어 알맞은 카드의 값을 GameController에 전달한다.

4.3 DB 구축

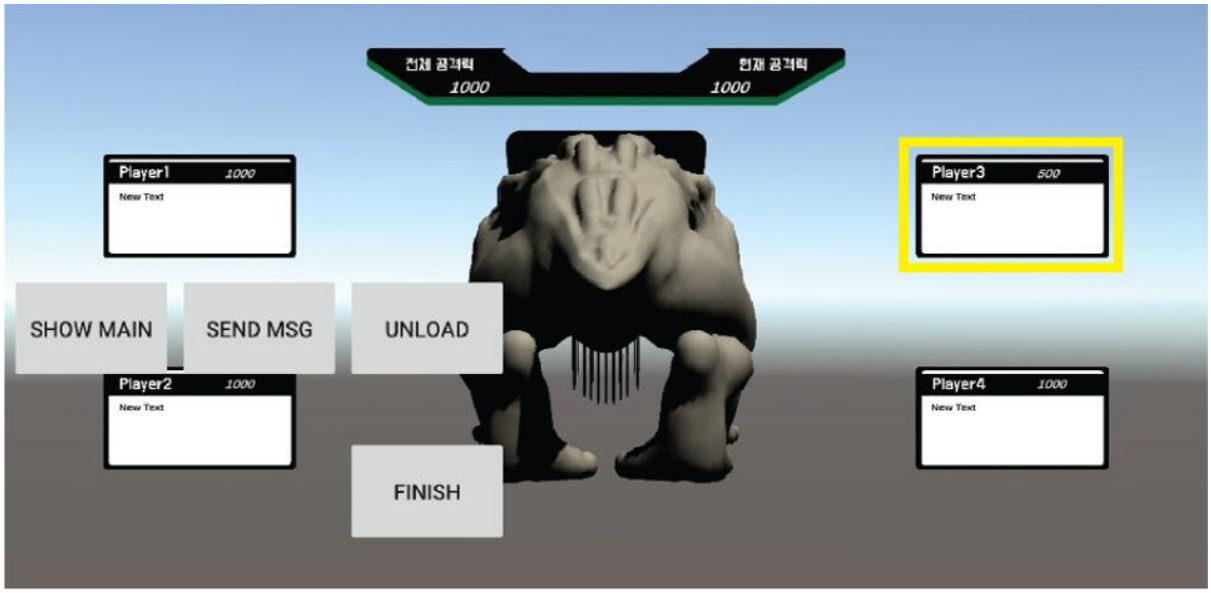
4.3.1) 카메라로 읽어온 카드 정보로 DB에 접근하여 올바른 데이터를 불러오는지 확인한다.

4.4 Unity

- 읽어온 카드 정보를 가지고 해당하는 3D Model을 찾는다.

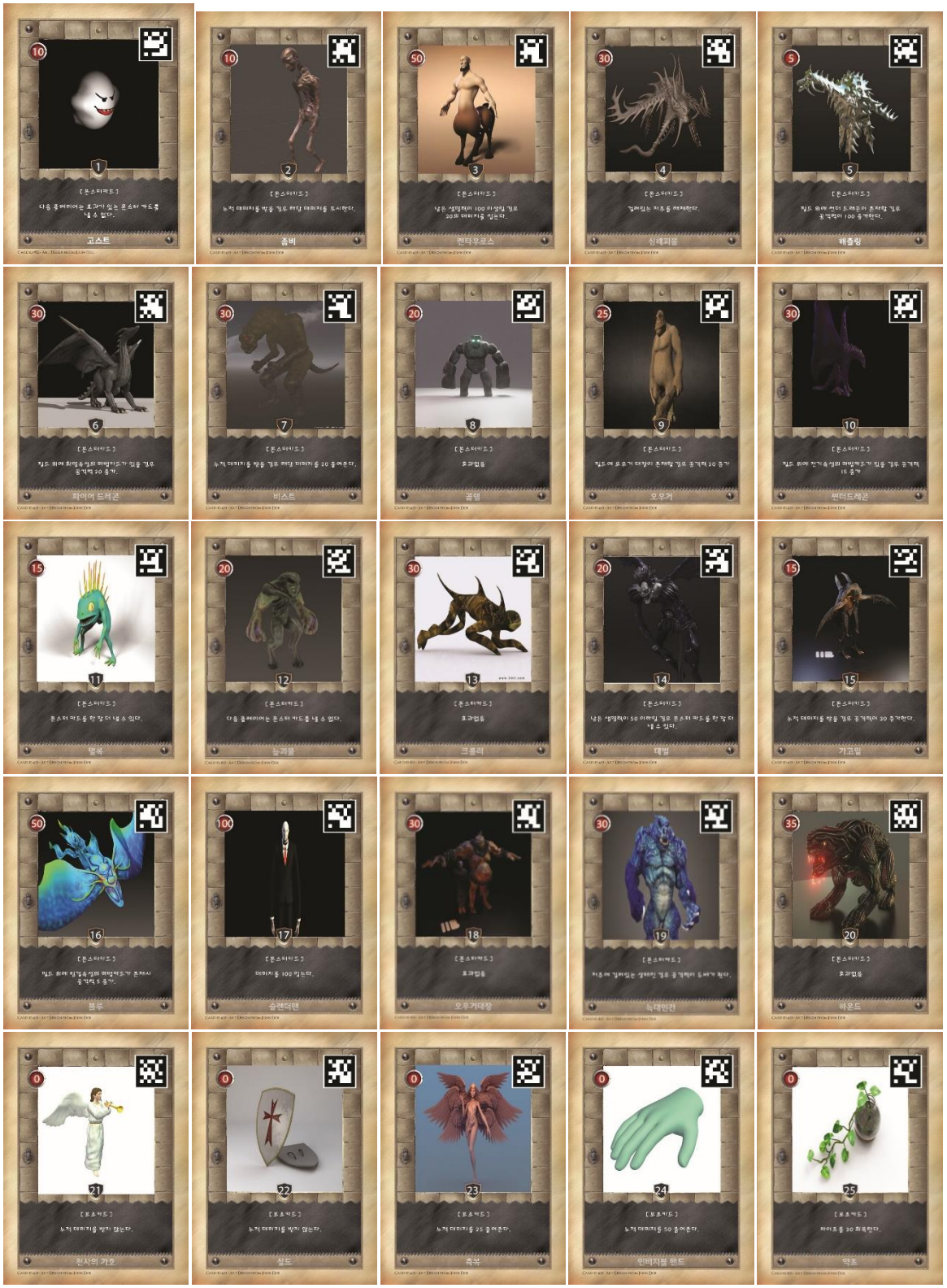


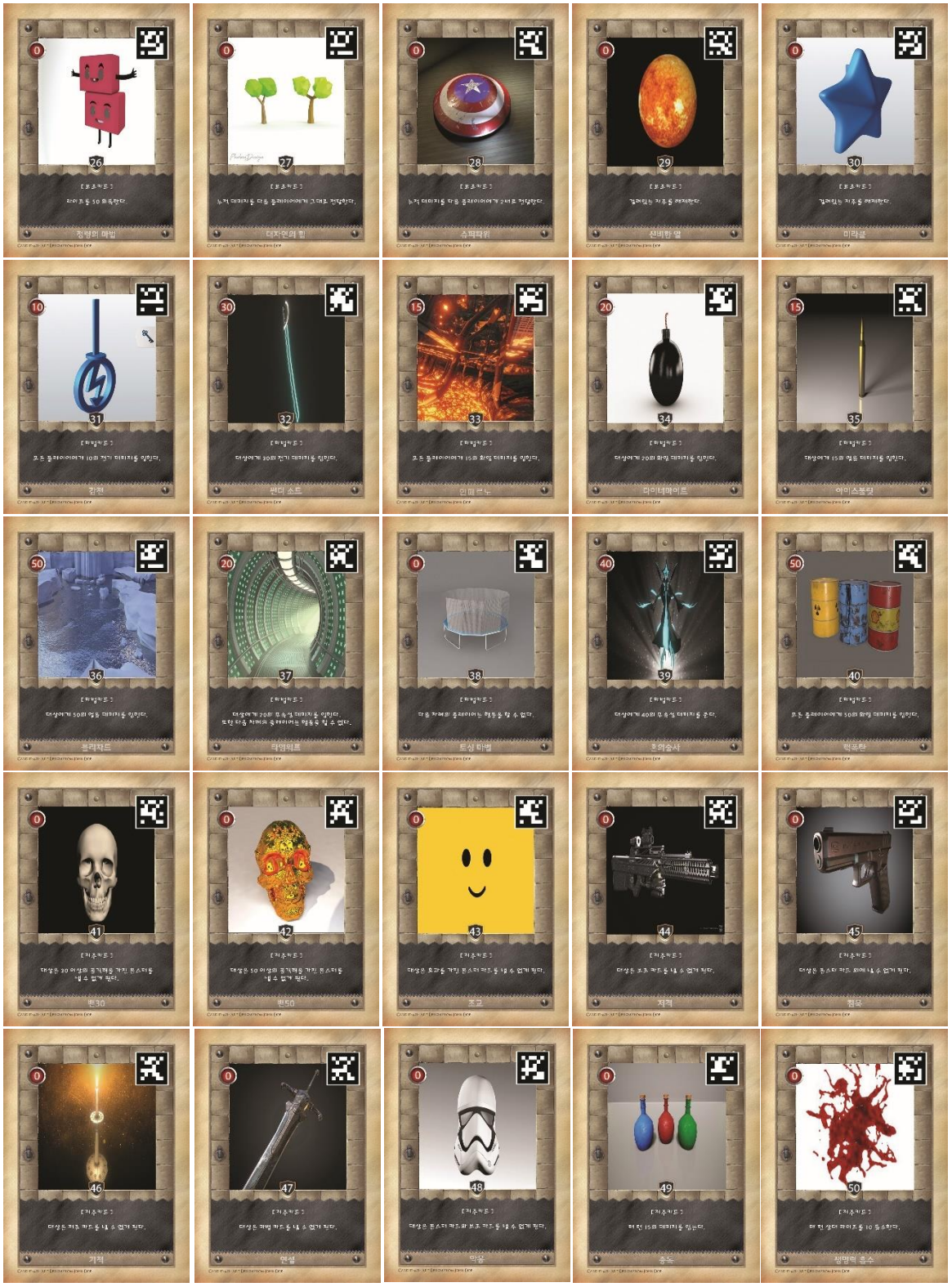
- 3D Model을 화면에 출력한다.



5 구현결과물

5.1 실물카드


10
1
[몬스터카드]
다음 플레이어는 효과가 있는 몬스터 카드를 낼 수 없다.
고스트
10
2
[몬스터카드]
능력 데미지를 받을 경우 해당 데미지를 도시한다.
좀비
50
3
[몬스터카드]
남은 생명력이 100 이상일 경우 20의 데미지를 입는다.
켄타우로스
30
4
[몬스터카드]
갖추고 있는 카드를 폐기한다.
성배괴물
5
5
[몬스터카드]
필드 위에 적이 드래곤이 존재할 경우 공격력이 100 증가한다.
배출왕
30
6
[몬스터카드]
필드 위에 화염속성의 마법카드가 있을 경우 공격력 20 증가.
파이어 드래곤
30
7
[몬스터카드]
능력 데미지를 받을 경우 해당 데미지를 20 줄여준다.
비스트
20
8
[몬스터카드]
효과없음
골렘
25
9
[몬스터카드]
필드에 오우거 타입이 존재할 경우 공격력 20 증가
오우거
30
10
[몬스터카드]
필드 위에 전기속성의 마법카드가 있을 경우 공격력 15 증가
썬더드래곤
15
11
[몬스터카드]
몬스터 카드를 한 장 더 낼 수 있다.
벽룩
20
12
[몬스터카드]
다음 플레이어는 몬스터 카드를 낼 수 없다.
늑대왕
30
13
[몬스터카드]
효과없음
크툴리
20
14
[몬스터카드]
남은 생명력이 50 이하일 경우 몬스터 카드를 한 장 더 낼 수 있다.
데빈
15
15
[몬스터카드]
능력 데미지를 받을 경우 공격력이 20 증가한다.
거고일
50
16
[몬스터카드]
필드 위에 빙결속성의 마법카드가 존재시 공격력 5 증가.
블루
100
17
[몬스터카드]
데미지를 100 준다.
슬랜더맨
30
18
[몬스터카드]
효과없음
오우거대장
30
19
[몬스터카드]
카드에 갖추고 있는 생명력 일부 공격력이 도배가 된다.
늑대인간
35
20
[몬스터카드]
효과없음
하운드
0
21
[포초카드]
능력 데미지를 받지 않는다.
천사의 가호
0
22
[포초카드]
능력 데미지를 받지 않는다.
실드
0
23
[포초카드]
능력 데미지를 25 줄여준다.
축복
0
24
[포초카드]
능력 데미지를 50 줄여준다.
인비지블 핸드
0
25
[포초카드]
라이프를 30 회복한다.
약초


0
26
[보조카드]
하이프를 50 회복한다.
천명의 매발
0
27
[보조카드]
눈적 데미지를 다음 플레이어에게 그대로 전달한다.
대자연의 힘
0
28
[보조카드]
눈적 데미지를 다음 플레이어에게 2배로 전달한다.
수박파워
0
29
[보조카드]
결투하는 지주를 해제한다.
신비한 알
0
30
[보조카드]
결투하는 지주를 해제한다.
미라클
10
31
[마법카드]
모든 플레이어에게 10의 전기 데미지를 입힌다.
감전
30
32
[마법카드]
대상에게 30의 전기 데미지를 입힌다.
번더 스트
15
33
[마법카드]
모든 플레이어에게 15의 화염 데미지를 입힌다.
인페르노
20
34
[마법카드]
대상에게 20의 화염 데미지를 입힌다.
파이널메이트
15
35
[마법카드]
대상에게 15의 힘을 데미지를 입힌다.
어미스투탐
50
36
[마법카드]
대상에게 50의 힘을 데미지를 입힌다.
블러처드
20
37
[마법카드]
대상에게 20의 독속성 데미지를 입힌다.
또한 다음 차례의 플레이어는 행동을 할 수 없다.
타임워프
0
38
[마법카드]
다음 차례의 플레이어는 행동을 할 수 없다.
토성 마법
40
39
[마법카드]
대상에게 40의 독속성 데미지를 준다.
흑역술사
50
40
[마법카드]
모든 플레이어에게 50의 화염 데미지를 입힌다.
핵폭탄
0
41
[지주카드]
대상은 30 이상의 공격력을 가진 몬스터를
내실 수 없게 된다.
뿐30
0
42
[지주카드]
대상은 50 이상의 공격력을 가진 몬스터를
내실 수 없게 된다.
뿐50
0
43
[지주카드]
대상은 호고를 가진 몬스터 카드를 내실 수 없게 된다.
조교
0
44
[지주카드]
대상은 포크 카드를 내실 수 없게 된다.
저격
0
45
[지주카드]
대상은 몬스터 카드 외에 내실 수 없게 된다.
함묵
0
46
[지주카드]
대상은 저주 카드를 내실 수 없게 된다.
거적
0
47
[지주카드]
대상은 마법 카드를 내실 수 없게 된다.
면설
0
48
[지주카드]
대상은 몬스터 카드와 포크 카드를 내실 수 없게 된다.
함묵
0
49
[지주카드]
하얀 15의 데미지를 입는다.
중독
0
50
[지주카드]
이 턴 상대 카드를 10 증가한다.
심명력 흡수

5.2 게임화면

6 Success Criteria

6.1 Test Case

Identifier	Feature	Valid Value
1.1	카드를 인식했을 경우 Card Recognition 모듈을 통해 코드를 해독한다.	화면에 실물영상이 계속 update되는것을 확인
2.1	카메라를 통해 캡쳐된 영상을 분석하여 해당 이미지에 코드데이터가 존재하는지 분석한다	캡처한 이미지의 Hex값 중 우리가 설정한 특정 색깔의 Hex값 이 일정 갯수 이상 존재하는지 확인
2.2	코드를 찾았을 경우 코드를 방향과 각도에 상관없이 데이터를 읽을 수 있게 보정처리 한다. 보정처리한 코드를 해독 모듈로 넘겨준다.	코드를 담고있는 이미지의 기울기 각도가 0 +- 5도 인지 확인
3	플레이어 라이프에 변동이 있거나, 차례가 바뀔 때 해당 정보를 Display controller에 update한다	Update 된 UI를 확인
4	읽어온 카드 데이터를 가지고 알맞은 행동을 취하며 게임을 진행시킨다.	플레이어 차례를 나타내는 글로벌 변수가 정상적으로 업데이트 되었는지 확인.
5	플레이어가 몬스터 카드를 냈을 경우, 해당 몬스터의 그래픽을 화면에 띄워준다.	Update 된 UI를 확인
6	코드를 해독해 해당 카드의 종류가 어떤 종류인지, (0: 몬스터카드, 1: 방어카드, 2: 회복카드, 3: 저주카드), 몇번 카드인지(1, 2, ...) 정수 데이터를 얻어낸다. 이후 데이터 파서로 해당 정수값이 전달된다.	코드 해독 결과 정수로 된 값이 2개 얻어지는지 확인.
7	해독된 정수 데이터를 가지고 DB에서 검색한 후 카드 데이터를 읽어온다.	카드의 정보를 담고있는 String 값을 얻었는지 확인.
8	복호화한 코드 데이터를 가지고 DB에서 카드의 정보를 읽어올 수 있다.	찾은 정보를 콘솔에 출력, 확인
9	카드를 신규 제작하였을때, 해당 카드 정보를 DB상에 추가할 수 있다.	등록한 정보를 콘솔 출력 시 정상출력 되는지 확인
10	사용자가 카드를 테이블에 내려놓은 순간부터 시작해서 카메라는 해당 카드를 3초 이내에 인식 완료하여야 한다.	타이머 3초 이내 확인
11	이 게임을 한번도 해보지 않은 플레이어도 화면을 보고 1초 이내에 수치를 읽을 수 있도록 라이프와 데미지의 UI요소들을 가시성과 가독성이 좋게 배치한다.	-

6.2 목표 달성률

6.2.1 1차 목표 달성률

Identifier	Feature	Valid Value	Iteration 1
1.1			
2.1	카메라를 통해 캡쳐된 영상을 분석하여 해당 이미지에 코드 태그가 존재하는지 분석한다	List<Contour> 중 사각형 영역 존재, 해당 사각형이 다른종류 사각형이 아닌 태그 사각형임	Pass
2.2	코드를 찾았을 경우 코드의 4가지 방향중 알맞은 방향으로 설정한 후 코드를 해독한다.	10bit의 Integer 값	Fail
3	플레이어 라이프나 상태에 변동이 생기는 경우, 변경된 정보로 업데이트해준다.	Update 된 UI를 확인	Fail 50%
4	읽어온 카드 데이터를 가지고 알맞은 행동을 취하며 게임을 진행시킨다. X - 미구현	플레이어 차례를 나타내는 글로벌 변수 업데이트 확인	Fail
5	플레이어가 실물 카드를 냈을 경우, 해당 카드의 그래픽을 화면에 띄워준다.	Update 된 UI를 확인	Fail
6	카드 정보를 읽어 어떤 종류의 카드인지 분석한다. X - 미구현	0~3 까지의 Integer Value	Fail
7	Integer 데이터를 이용해 데이터베이스에서 카드 정보를 읽어온다.	카드의 정보를 담은 data Class를 얻었는지 확인	Fail 50%
8			
9	카드를 신규 제작하였을때, 해당 카드 정보를 DB상에 추가할 수 있고, 중복된 데이터는 실패처리한다.	콘솔 로그 메시지로 db 쿼리 수행결과 확인	Pass
10	사용자가 카드를 테이블에 내려놓은 순간부터 시작해서 카메라는 해당 카드를 3초 이내에 인식 완료하여야 한다.	타이머 경과시간 <= 3sec	Fail
11	이 게임을 한번도 해보지 않은 플레이어도 화면을 보고 1초 이내에 수치를 읽을 수 있도록 라이프와 데미지의 UI요소들을 가시성과 가독성이 좋게 배치한다.	-	Pass

6.2.2 2차 목표 달성률

Identifier	Feature	Valid Value	Iteration 2
1.1			
2.1	카메라를 통해 캡쳐된 영상을 분석하여 해당 이미지에 코드 태그가 존재하는지 분석한다	List<Contour> 중 사각형 영역 존재, 해당 사각형이 다른종류 사각형이 아닌 태그 사각형임.	Pass
2.2	코드를 찾았을 경우 코드의 4가지 방향중 알맞은 방향으로 설정한 후 코드를 해독한다.	10bit의 Integer 값	Pass
3	플레이어 라이프나 상태에 변동이 생기는 경우, 변경된 정보로 업데이트해준다.	Update 된 UI를 확인	Fail 50%
4	읽어온 카드 데이터를 가지고 알맞은 행동을 취하며 게임을 진행시킨다. X - 미구현	플레이어 차례를 나타내는 글로벌 변수 업데이트 확인	Fail 50%
5	플레이어가 실물 카드를 냈을 경우, 해당 카드의 그래픽을 화면에 띄워준다.	Update 된 UI를 확인	Fail 50%
6	카드 정보를 읽어 어떤 종류의 카드인지 분석한다. X - 미구현	0~3 까지의 Integer Value	Fail 50%
7	Integer 데이터를 이용해 데이터베이스에서 카드 정보를 읽어온다.	카드의 정보를 담은 data Class를 얻었는지 확인	Fail 50%
8			
9	카드를 신규 제작하였을때, 해당 카드 정보를 DB상에 추가할 수 있고, 중복된 데이터는 실패처리한다.	콘솔 로그 메시지로 db 쿼리 수행결과 확인	Pass
10	사용자가 카드를 테이블에 내려놓은 순간부터 시작해서 카메라는 해당 카드를 3초 이내에 인식 완료하여야 한다.	타이머 경과시간 <= 3sec	Fail
11	이 게임을 한번도 해보지 않은 플레이어도 화면을 보고 1초 이내에 수치를 읽을 수 있도록 라이프와 데미지의 UI요소들을 가시성과 가독성이 좋게 배치한다.	-	Pass

6.2.3 최종 목표 달성률

7. 게임 규칙

◆ 카드 구성

- 이 게임은 몬스터카드 20장, 보조카드 10장, 마법카드 10장, 저주카드 10장으로 구성됩니다.

<카드정보>

★ 몬스터 카드

- 배틀의 주역!

▶ 효과를 가지지 않은 '일반 몬스터'

▶ 특수한 효과를 가진 '효과 몬스터'

★ 보조 카드

- 데미지를 최소화 할 수 있는 카드

★ 마법 카드

- 발동하면 여러 효과를 발휘할 수 있는 카드

★ 저주 카드

- 다른 플레이어의 행동을 제한할 수 있는 카드

◆ 게임 설명

- 덱에서 소환한 배틀카드로 상대를 공격하여 본인을 제외한 모든 플레이어의 생명력을 0으로 만들고 마지막 남은 최후의 플레이어가 승자가 되는 게임이다.

◆ 게임 순서

1. 카드를 5장씩 나누어 가진다.

나머지 카드더미는 섞은 후 뒷면이 보이게 해서 가운데 쌓아둔다.

2. 자기 차례가 오면 손에 있는 카드 중에 필드 위에 조건에 맞는 카드를 내고 카드더미에서 한 장을 가져올 수 있다.

3. 카드에 적힌 효과에 따라 게임을 진행하고 마지막에 남은 사람이 승자가 된다.

8. 향후 계획

본 프로젝트에서는 AR Card Game을 자체적인 2D Code와 Game Logic에 따라 모바일 어플리케이션으로 구현하였다. Unity와 메시지를 지속적으로 주고받으며 원하는 게임로직을 동작시킴과 동시에 그래픽을 나타내었다. 이와 같은 방식과 더불어 직접 퀄리티 높은 3D Model을 디자인하여 개발한다면 기존의 게임보다 더욱 현실감 있고 재밌는 카드게임을 구현할 수 있을 것으로 기대된다.