프로젝트 최종보고서

1675022 김현지 / 1675047 이소현

목차

1. 설계 주제 및 목표

서점에 가면 성인들은 읽고 싶은 책을 찾기 위해 인터넷 검색을 하거나 아예 자리에 앉아 책의 전반부를 읽어보기도 한다. 그러나 어린 아이들의 경우 직접 읽고 싶은 책을 검색하기 쉽지 않다. 아동도서의 간단한 검색기능을 구현할 수 있는 방법은 없을까? 요즘 공공장소에서 울고 있는 아이들에게 스마트폰이나 태블릿 PC 를 쥐어주는 일이 흔해졌다. 이처럼 어렸을 때부터 디지털기기와 친숙한 아이들이 간단한 터치만으로 어플리케이션을 이용할 수 있다면 직접 읽고싶은 책을 고를 수 있을 뿐만 아니라 책에 대한 흥미도 유발할 수 있을 것이다. 이러한 생각에 기반하여 핸드폰으로 책을 촬영하면 해당 책의 예고편이나 후킹 메시지를 띄워 주는 기능을 내장하고 있는 어플리케이션을 제작해보기로 하였다. 이러한 아이디어는 추천하고 싶은 도서의 카드뉴스를 제작하는 '책 끝을 접다'라는 페이지에서 영감을 얻어, 이를 아동도서에 적용하고 그 자체로 흥미를 유발할 수 있도록 AR(증강현실)을 사용하고자 한다. 설계목표는 책 표지를 촬영하여 표지의 텍스트롤 책 제목으로 인식하고 해당하는 AR 영상을 책 표지 이미지위에 송출하는 것이다.

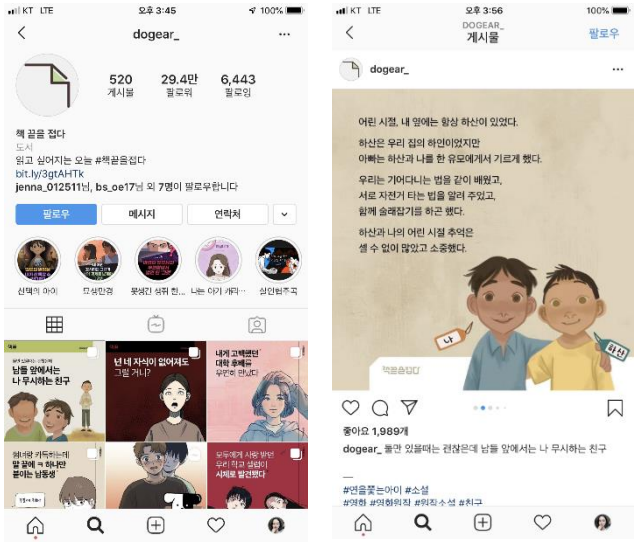


그림 1 '책 끝을 접다' 페이지와 카드뉴스

2. 기존 문헌 분석, 수행 및 새로운 아이디어 제안

1) 기존 문헌 분석 및 수행 -ARCore Augmented Images code

https://codelabs.developers.google.com/codelabs/augimg-intro/index.html?index=..%2F..index#6

공간기반 AR(증강현실)을 구현하기 위한 소프트웨어 개발 키트는 대표적으로 Apple 의 ARKit, Google 의 ARCore 두 가지가 존재한다. 이 중에서 AR 어플리케이션 제작을 위해 ARCore 를

이용하였다. Google Codelab 에서 제공하는 기존 ARCore Augmented Images 코드를 기반으로 하여 원하는 방향으로 수정 작업을 진행하였다. 먼저, ARCore Augmented Images 코드를 이용하여 안드로이드 스튜디오의 기본적인 사용법을 익히고 이미지 파일의 네 모서리에 frame

그림 2~3 기존 AR object와 새로 import한 AR maze object를 기존 이미지위에 송출한 모습

image 를 올려서 image detection 이 제대로 이루어졌는지 확인한다. 기존 코드에서는 내제되어있는 한 이미지에 코드에서 설정한 한 ARobject 만을 띄울 수 있었다.
그 이미지 위에 임의의 object(기존 코드에서는 3d maze 이미지)를 쌓아올렸다. 3D model 을 원하는 형태로 송출하기 위해서는 3D object 파일을 불러와서 이를 sfa, sfb 파일로 각각 저장하고 sfa 파일에서 scale 이나 roughness 값을 조정해야 한다. 기존 코드를 이용하면 앱을 켜고 직접 촬영한 이미지를 Anchor 으로 지정할 수 있다.

2) 새로운 아이디어 제안

본 프로젝트에서는 기존 연구에서 다음과 같은 부분을 발전시켜 주제에 맞는 어플리케이션을 제작하고자 한다.

1. 다양한 3D object 를 rendering 하는 것
2. 예고편으로 사용할 간단한 AR 영상을 제작하는 것
3. 책 제목과 일치하는 AR 영상을 송출하는 것

결과적으로, 어플리케이션을 실행하여 검색하고자 하는 책의 이미지를 촬영하면, 어플리케이션이 책의 제목을 인식하여 바닥 이미지가 지정되고, 그 위에 그에 맞는 3d 이미지를 순차적으로 보여주는 예고편 형식의 어플리케이션을 제작 할 것이다.

3. Block diagram

1) 글씨인식 in FirstActivity

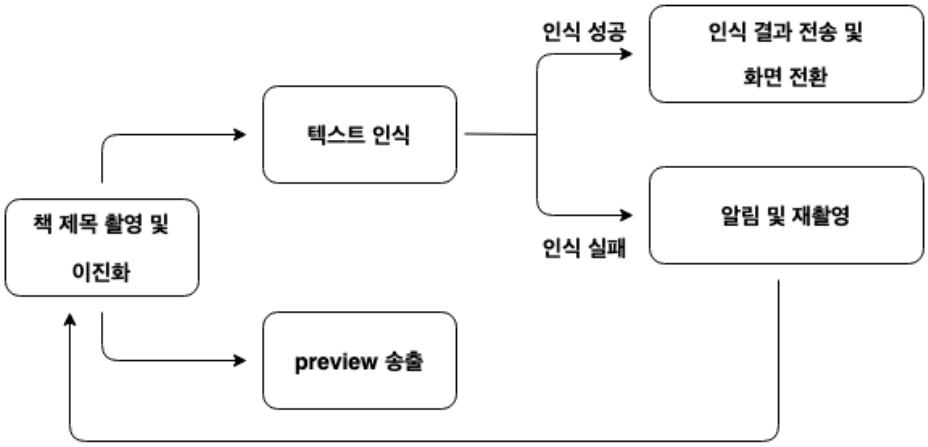


그림 4 FirstActivity의 Block diagram

첫 번째 단계는 어플리케이션을 실행한 후, 핸드폰 카메라로 책을 촬영하여 제목을 인식하는 단계이다. 어플리케이션을 실행하면 자동으로 책 표지 이미지를 인식하여 이진화하는 과정이 이루어져야 한다. 이진화 과정을 거친 이미지는 흑백 영상이 되어 black 영역을 텍스트로 인식하고 아래 작은 화면으로 preview 를 송출함과 동시에 tesseract 를 이용하여 제목을 추출한다. 텍스트 인식에 성공하면 인식 결과를 Image Augmented Activity 의 입력값으로 전송하고, 텍스트 인식에 실패하면 실패 알림을 띄우고 재촬영 화면으로 넘어간다.

2) AR in ImageAugmentedActivity

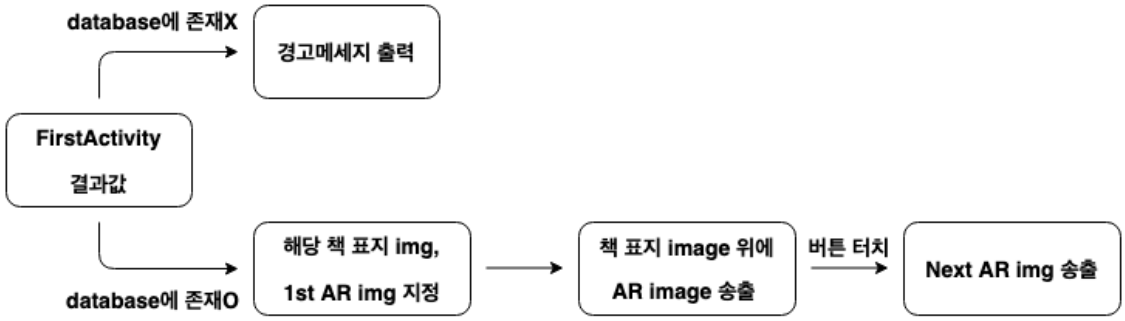


그림 5 ImageAugmentedActivity의 Block diagram

두 번째 단계는 책 표지를 인식하여 얻은 결과값(책 제목)으로 AR 영상을 송출하는 단계이다. AR 영상을 송출하기 위하여 안드로이드 스튜디오를 기반으로 제작한 AR 어플리케이션을 이용한다. FirstActivity 의 결과값을 기존 database 에서 찾아 일치하는 항목이 없는 경우에는 경고메세지를 출력하고 일치하는 항목이 있는 경우에는 AR 영상을 송출하기 위한 해당 책 표지 이미지와 첫 번째로 띄울 AR 영상을 변수로 지정한다. 지정이 완료되면 책 표지 image 위에 AR 영상을 송출하고 버튼을 누르면 다음 AR image 를 송출한다.

3) 전체 block diagram

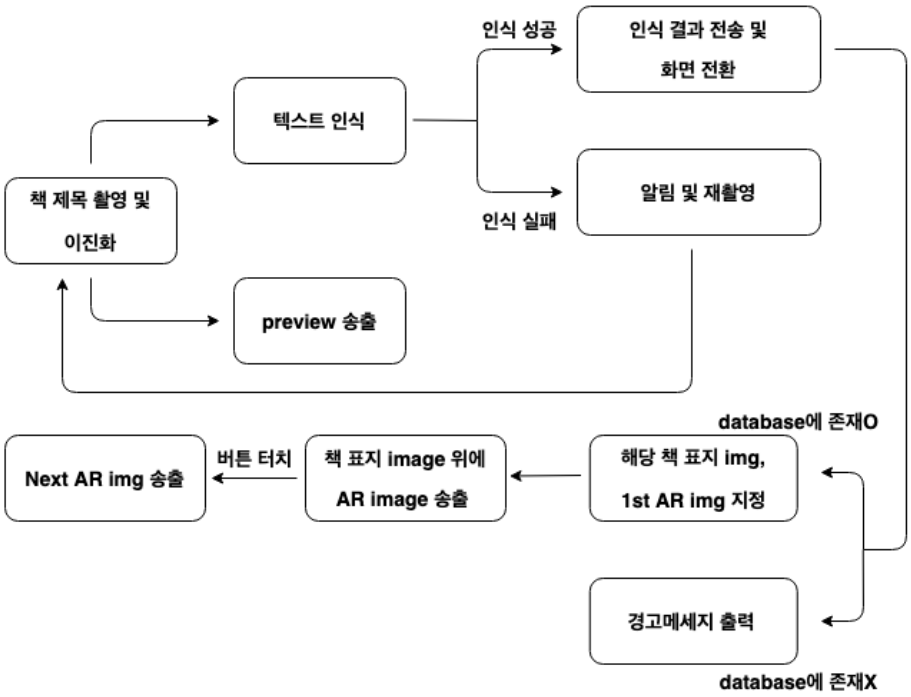


그림 6 어플리케이션 전체의 Block diagram

위에서 설명한 내용을 전체적인 하나의 block diagram 으로 표현하면 다음과 같다. 어플리케이션이 정상적으로 작동하기 위해서는 크게 6 단계를 거쳐야한다. 핸드폰의 application 을 실행하여 책 표지를 촬영하면 FirstActivity 코드에서 표지를 이진화하여 책 제목에 해당하는 텍스트 부분만을 추출하고 database 에서 결과와 일치하는 항목을 찾아 책 표지 image 위에 사전에 database 로 입력해놓은 AR 영상을 차례대로 송출한다. 만약 텍스트 인식이 제대로 이루어지지 않거나 database 에 제목과 일치하는 책이 없는 경우 경고메세지를 출력한 후 이전 단계로 돌아가도록 하는 feedback 과정이 이루어진다.

4. 동작 원리 및 구현 전 데이터 처리

1) 글씨인식 OCR - Tesseract, trained data 제작

이진화 과정에서 black 색상으로 이진화된 텍스트는 tesseract OCR 로 추출한다. Tesseract 는 다양한 운영 체제를 위한 광학 문자 인식 엔진으로 input 이미지의 특징점을 추출하고 그 특징점을 사용하여 문자를 인식하는 문자 인식 기법이다. Tesseract 가 문자의 특징점을 추출하는 방법은 다음과 같다. 우선 input image 의 outline 을 생성하고 방향성을 정한다. 그리고 그 방향성에 따라 다각형 모양을 추출한다. 특징점을 추출하고 나면 Tesseract 의 database 검색을 통해 특징점이 비슷한, 즉 오차율이 가장 낮은 문자를 찾는다.

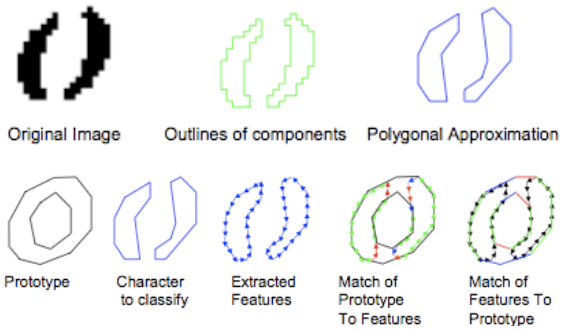


그림 7 Tesseract의 동작 원리

.

안드로이드 스튜디오에서 Tesseract-OCR 을 사용하기 위해서는 Tesseract 로 training 한 trained data 가 필요했다. 이를 위해 메모장에서 다양한 폰트의 알파벳을 작성하고 tiff 로 저장한 뒤 tesseract 의 box making 을 통해 box 파일을 만들어 주었다. Tiff 와 box 파일을 연동하여 tesseract 프로그램에서 trained data 를 제작했다.

2) AR sceneform

증강현실은 (Augmented reality; AR) 은 가상현실(VR)의 한 분야로 실제로 존재하는 환경에 가상의 사물이나 정보를 합성하여 마치 원래의 환경에 존재하는 사물처럼 보이도록 하는 컴퓨터 그래픽 기법이다.

본 프로젝트의 AR 은 ARCore 의 sceneform 을 이용하여 구현하였다. Sceneform 이란 ARCore 에서 기술 장벽이 높은 OpenGL 사용 없이도 3D scene 을 rendering 할 수 있게 구글에서 제공하는 API 이다. Physically based rendering 을 이용하여 높은 퀄리티의 렌더링을 할 수 있게 해준다. AR sceneform 의 AR 화면 구성요소는 다음과 같다.

- Scene : AR 화면을 의미한다.
- Anchor : 3D object 를 놓을 수 있는 평면의 구성요소이다. 여러개의 Anchor 가 모여 오브젝트가 놓일 수 있는 공간을 만든다. 본 프로젝트에서는 원하는 이미지로 Anchor 를 설정하기 위해 ArFragment 로 구성을 확장해주었다.
- Renderable: Node 영역 안에 있는 오브젝트를 의미한다. sfb 파일로 source 를 지정해 준다.
- Node : 하나의 오브젝트가 차지하는 영역을 의미한다.

(1) The arcoreimg tool

Arcoreimg는 AR이미지를 보여줄 바닥 Anchor이 될 이미지의 퀄리티를 계산하고 이미지 데이터 베이스를 만드는 툴이다. AR의 Anchor가 되기 위해서는 이미지가 다른 이미지와 명확히 구분되는 특징을 가지고 있어야 한다. Arcoreimg는 각각의 이미지들이 Anchor에 적합한지 0에서 100까지 점수를 매겨준다. 75점 이상이 Anchor가 되기에 적합한 이미지이다. Arcoreimg로 바닥이 될 image의 database를 제작해 주었다. 75점 이상의 이미지가 담긴 아래와 같은 방식으로 이미지의 path를 담은 text파일을 input 으로 지정하여 .imagedb

파일을 build해 주었다.

```
A fox and a crane|D:\H\EWHA\4-1\DIP\Project\HJ\sceneform-android-sdk-
1.9.0\samples\augmentedimage\app\src\main\assets\A fox and a crane.jpg
```

```
C:₩>arcoreimg.exe build-db --input_image_list_path=D:₩H₩EWHA₩4-1₩DIP₩Project₩HJ₩sceneform-a
ndroid-sdk-1.9.0₩samples₩augmentedimage₩app₩src₩main₩assets₩imglist.txt --output_db_path=D:
₩H₩EWHA₩4-1₩DIP₩Project₩HJ₩sceneform-android-sdk-1.9.0₩samples₩augmentedimage₩app₩src₩main₩
assets₩images.imgdb
Image database generated at: D:₩H₩EWHA₩4-1₩DIP₩Project₩HJ₩sceneform-android-sdk-1.9.0₩sampl
es₩augmentedimage₩app₩src₩main₩assets₩images.imgdb
Image list file generated at: D:₩H₩EWHA₩4-1₩DIP₩Project₩HJ₩sceneform-android-sdk-1.9.0₩samp
les₩augmentedimage₩app₩src₩main₩assets₩images.imgdb-imglist.txt
```

그림 8 arcoreimg를 이용해 data base를 build함

(2) AR 3D File export

tesseract OCR 로 얻어낸 결과값을 기반으로 AR Scene 을 송출하기 위해 AR 3D file 을 제작해 두었다. AR 3D file 은 poly 와 unity 를 사용하여 제작하였다. AR content 로 사용할 다양한 3d 이미지를 얻기 위해 poly 를 이용하였다. Poly 는 구글에서 사람들이 tilt brush 나 block 으로 제작한 콘텐츠를 무료로 사용할 수 있도록 해주는 AR, VR 개발을 위한 3d contets 공유 플랫폼이다. Poly 이미지는 다양한 파일 형식으로 제공되는데 안드로이드 스튜디오에서 3D contents 를 .sfa 와 .sfb 파일로 export 하기 위해서는 3D file(.obj, .fbx, .gITF)을 사용해야 한다.

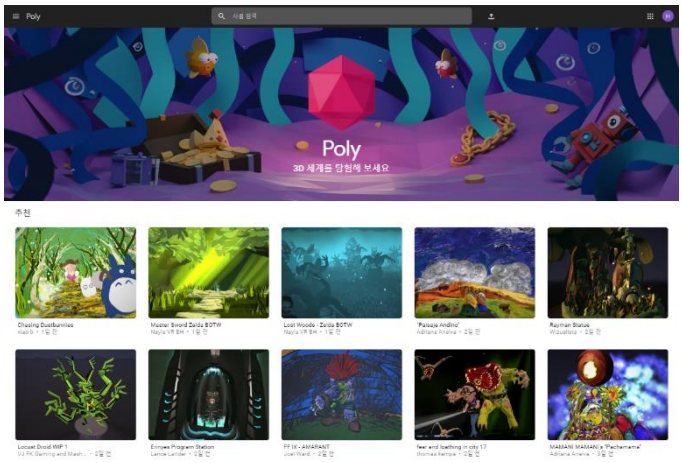


그림 9 3D model 공유 platform poly

Poly asset 을 import 하기 위해서는 unity 에서 poly toolkit 을 사용해야 한다. Unity 는 3D 및 2D 비디오 게임의 개발 환경을 제공하는 게임 엔진이자, 3D 애니메이션과 건축 시각화, 가상현실(VR) 등 인터랙티브 콘텐츠 제작을 위한 통합 제작 도구이다. 3d poly 이미지와 3d 텍스트 simple helvetica 등 다양한 AR 자료를 이용하기 위해서는 unity 의 asset 을 활용할 수 있다. 따라서 asset store 에서 poly toolkit 와 simple helvetica asset 을 다운받아 이미지나 텍스트를 원하는 크기로 scaling 하거나 위치를 지정하고 해당 이미지를 원하는 형식(.fbx, .obj)으로 export 하였다.

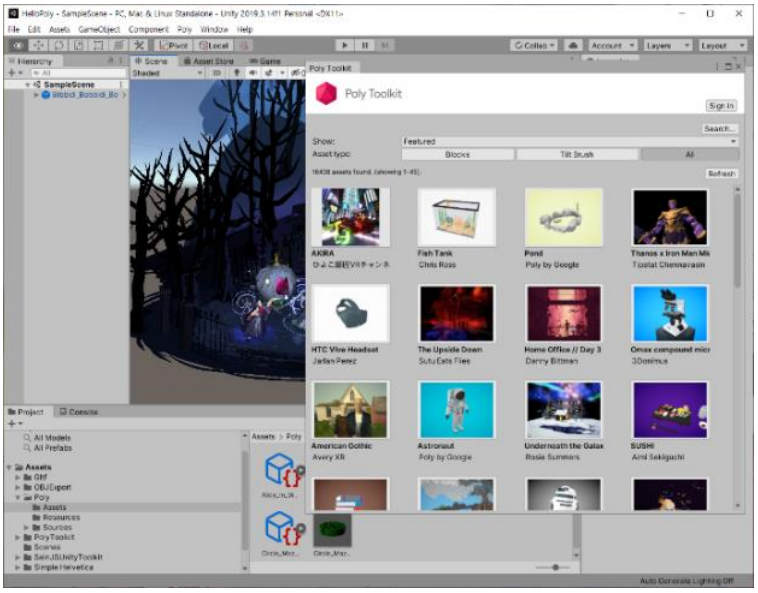


그림 10 Unity에서 Poly toolkit을 사용하는 모습

5. 어플리케이션 구현-source code

어플리케이션은 크게 그림과 AR 파일들로 구성된 데이터와 xml 레이아웃 파일, Java class 파일들로 구성된다. 어플리케이션의 동작은 Java class 파일에서 구현된다. 자바는 객체 지향 언어이기 때문에, 각 class 별로 구성 요소를 가져올 수 있다. 해당 어플리케이션을 구현하기 위해 5 가지의 class 를 만들어 주었다. Java class 파일에는 글씨 인식을 위한 FirstActivity, FirstActivity 에서 휴대폰 카메라를 이용하기 위한 CameraSurfaceView 가 있고, AR 을 보여주기 위한 AugmentedImageActivity, 바닥이 될 Fragment 를 구성하는 AugmentedImageFragment, Fragment 위에 Anchor 를 구성하고 그 위에 보일 AR 을 Node 로 구성하여 rendering 하는 AugmentedImageNode 가 있다.

이 중 화면에 직접적으로 송출되는 FirstActivity 와 AugmentedImageActivity 를 중심으로 어플리케이션 구현을 설명할 것이다.

1) FirstActivity

(1) Region of Interest

본 팀이 사용할 책 이미지에는 책 제목과 그림이 존재했다. Tesseract input image 에 촬영한 이미지를 그대로 넣을 경우 해당 이미지안의 그림이 글씨로 잘못 인식 되기 때문에 글씨 부분만을 관심 영역(Region of Interest, ROI)로 지정하여 ROI 영역을 crop 한 뒤 해당 영역을 bitmap 으로 return 처리 해주었다.

(2) 책 표지 이미지 이진화

Tesseract 는 주변의 명암과 작은 에러에도 민감하게 반응한다는 단점이 있었다. 이를 위해 간단한 이진화 전처리를 거쳐 OCR 의 정확도를 높여주었다. 책 표지 이미지를 이진화하기 위해서 단순히 threshold 를 이용하는 것이 아닌 color distance 를 이용했다. 먼저, 인식된 책 표지 이미지 내부에 위치한 각 픽셀의 color 값을 color space 에서 (r,g,b)로 좌표화한다. Color space 는 8bit 를 사용하기 때문에 black 좌표는 (0,0,0), white 좌표는 (255,255,255)로 표현된다. 그리고 각 pixel 의 r,g,b 좌표값과 white 좌표값까지의 거리를 구한다. 그리고 white(255,255,255)와 black(0,0,0)까지의 거리 중 가까운 값으로 pixel 값을 binary 화 한다. 예를 들어 배경색의 r,g,b 성분이 (180,200,200), 글자색의 r,g,b 성분이 (30,30,30)라고 하자. 이 때, 배경색의 좌표값은 white 좌표까지의 거리가 약 108, black 좌표까지의 거리가 약 335 이므로 흰색으로 이진화되고, 글자색의 좌표값은 white 좌표까지의 거리가 약 389, black 좌표까지의 거리가 약 51 이므로 검정색으로 이진화 될 것이다. 이후에 Tesseract 를 이용하면 이진화 결과 black 색상으로 이진화된 부분을 글씨로 인식하고 텍스트가 추출될 것이다.

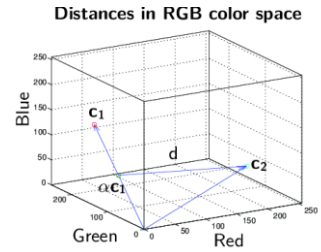


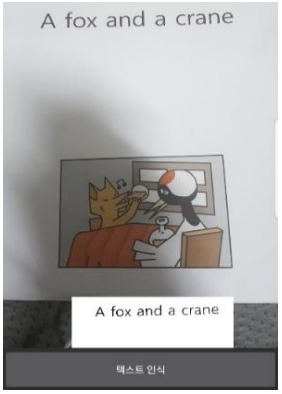
그림 11 Color distance in RGB

(3) OCR

촬영한 책 image 를 ROI crop 과 GetBinaryBitmap 함수를 거쳐 input image 로 사용했다. 이후 tesseract traineddata 를 담고있는 TessBaseAPI 를 이용하여 이미지를 text 로 반환시켜주고 그 결과를 String BookName 에 저장한 후 알림 창을 띄워 사용자가 책 제목이 맞는지 확인시켜주었다. 책 이름이 맞을 경우 Intent 를 이용해 AugmentedImage Activity 로 화면을 전환한다. Intent 는 다른 액티비티 간의 데이터를 주고 받기 위한 용도로 사용되는 객체이다.

```
tessBaseAPI = new TessBaseAPI();
String dir = getFilesDir() + "/tesseract";
if (checkLanguageFile( dir: dir + "/tessdata"))
    tessBaseAPI.init(dir,  language: "eng");
```

Source code 1 Tesseract data 를 이용해 tessBaseAPI에 넣어주는 코드



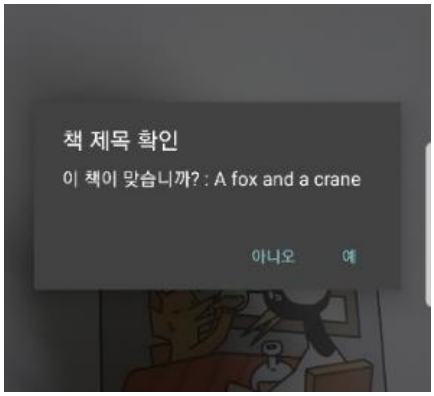


그림 12 (좌) 원본 촬영 화면과 아래 ROI crop및 이진화 된 이미지

그림 13 (우) 책 인식 결과 알림창. 예를 누르면 화면이 전환된다.

2) AugmentedImageActivity

AugmentedImageActivity 로 화면이 전환되면 가장 먼저 BookName 결과값을 이름으로 가지고 있는 이미지를 augmentedimage 로 지정해준다. 이때 해당 책의 이미지가 image database 에 없을 경우, "다음 책이 존재 하지 않습니다. -{BookName}"이라는 경고 알람을 띄운다. 해당 책이 database 에 존재 할 시 augmentedimage 를 Anchornode(Anchor 가 차지하는 영역)로 삼는다.

Augmentedimage 의 tracking 이 가능 할 때 AugmentedImageNode class 인 node 를 선언한 뒤, node 의 anchor 를 augmentedimage 로 지정해 주었다.

```
case TRACKING:
  // Have to switch to UI Thread to update View.
    fitToScanView.setVisibility(View.GONE);
    // Create a new anchor for newly found images.
  if (!augmentedImageMap.containsKey(augmentedImage)) {
    AugmentedImageNode node = new AugmentedImageNode( context: this);
    node.setImage(augmentedImage);
    augmentedImageMap.replace(augmentedImage, node);
    arFragment.getArSceneView().getScene().addChild(node);
  }
```

Source code 2 AugmentedImageActivity에서 augmentedimage가 tracking이 가능할 때 작동하는 코드

이후 arFragment(AR 을 송출할 장면)의 scene child 로 node 를 지정해 주었다. 이를 통해 해당 장면에 해당 node 가 송출 될 수 있다. 본 프로젝트에서는 node 를 늘리기 위해 화면의 버튼이 눌릴때마다 node 의 Renderable 을 바꾸고 다시 switch 문 안에서 arFragment 의 child 를 추가해 줌으로써, 한 화면에 여러개의 AR object 가 뜰 수 있게 하였다.

3 이는 node.setImage() 에서 구현된다

```
setAnchor(image.createAnchor(image.getCenterPose()));
mazeNode = new Node();
mazeNode.setParent(this);
mazeNode.setRenderable(mazeRenderable0.getNow( valueIfAbsent: null));

switch(((AugmentedImageActivity) AugmentedImageActivity.context_main).butNum){
    case 1:
        mazeNode.setRenderable(mazeRenderable1.getNow( valueIfAbsent: null));
        mazeNode.setLocalPosition(new Vector3( v: 0, v1: 0.07f,  v2: 0));
        break;
    case 2:
        mazeNode.setRenderable(mazeRenderable1.getNow( valueIfAbsent: null));
        mazeNode.setLocalPosition(new Vector3( v: 0.01f, v1: 0,  v2: 0));
        break;
    case 3:
        mazeNode.setRenderable(mazeRenderable1.getNow( valueIfAbsent: null));
        mazeNode.setLocalPosition(new Vector3( v: 0.01f, v1: 0.07f,  v2: 0));
        break;
}
```

Source code 3 AugmentedImageNode class안의 setImage 함수의 일부분

AugmentedImageNode class 파일의 setImage 함수 안에서 augmented image(해당 함수의 지역변수는 image) 정 중앙을 Anchor 로 지정해 준 뒤 node 를 생성하여 node 의

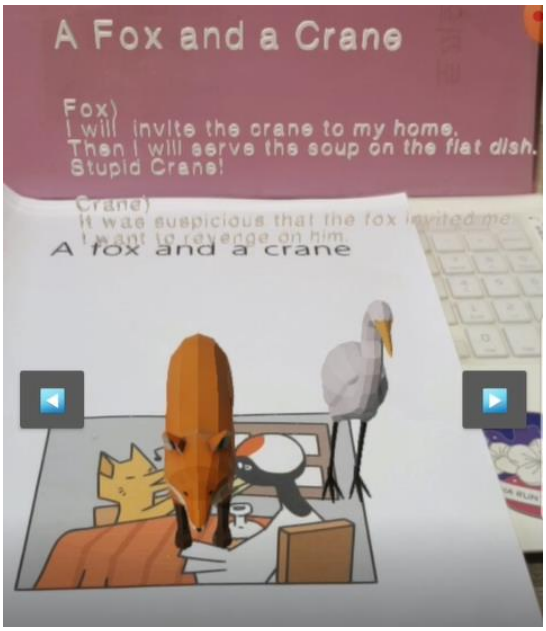
Renderable을 지정하고, node의 Position을 3차원 vector를 통해 지정해준다. 이때 v는 좌우, v1 은 위 아래, v2 는 앞 뒤 방향의 vector 이다.
AugmentedImageActivity 의 Button 을 누르면 butNum 의 숫자가 하나씩 올라가며, 그럴때마다 node 의 Renderable 을 재설정 해주었다. 해당 값을 사용하기 위해 context 를 사용했다.

```
mazeRenderable0 =
        ModelRenderable.builder()
                .setSource(context, Uri.parse(((FirstActivity) FirstActivity.context_first).BookName + "0.sfb"))
                .build();
```

Source code 3 AugmentedImageNode class안의 Renderable build 부분

이때 각각의 Renderable 은 "{BookName}{number}.sfb"를 source 로 build 되었으며, FirstActivity 의 결과값을 가져오기 위해 context 를 사용해 주었다.
Source 의 .sfb 파일을 만들기 위해 안드로이드 프로젝트 파일을 build 할때의 정보를 담고 있는 build.gradle 파일에서 sceneform asset 을 사용하였다. 3D format file(.obj, .fbx, .gltf) 은 sceneform.asset 을 통해 Sceneform Asset Definition file(.sfa)파일과 Sceneform Binary asset(.sfb)파일로 export 된다. 이때 .sfa 파일은 이 3D format file 의 정보를 담고 있기에, fragment 에 송출될 Renderable 의 scale 이나 material 등의 정보를 수정할 수 있다.

AugmentedImageActivity 에서 버튼을 누를때마다 arFragment 의 child 에 node 가 추가되고, 이는 frame 이라는 변수에 update 된다. 이후 Activity 의 화면에서 updateFrame 을 통해 AR 이 송출 된다.



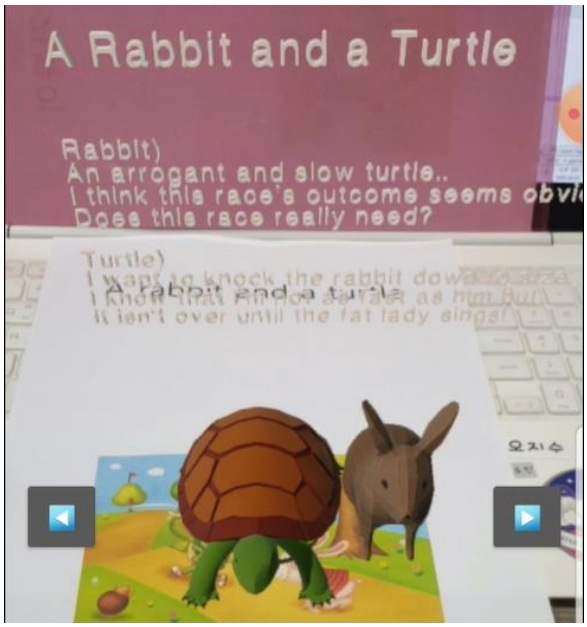


그림 14~15 책 제목에 따라 다르게 나오는 AR object들의 모습. 버튼을 끝까지 눌러 모든 AR object가 송출된 Fragment 상태이다.

6. 성능비교 및 개선한 점

1) Tesseract training

Tesseract 는 인식률 향상을 위해서 training 데이터를 적용할 수 있다. Training 과정을 거친 데이터들은 tesseract database 에 저장되어 인식한 글자와 training data 와의 유사율을 비교하여 인식 결과를 반환한다. Training data 를 구축하기 위해서는 다음과 같은 방법을 적용하였다.

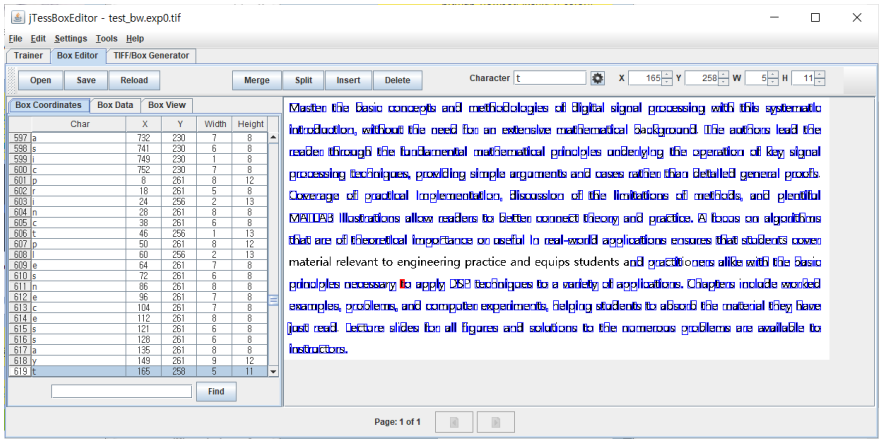


그림 4 box generator가 설정한 box 영역을 재설정하거나 잘못 인식한 철자를 교정하는 과정

우선 training 이미지 파일을 생성하고 이 파일로부터 box 파일을 생성한다. 이 때, 'tesseract.exe custoKOR.font.exp0.tif custoKOR.font.exp0 batch.nochop makebox' 명령어를 수행하면 tesseract OCR 모듈이 인식단위인 box 를 자동으로 인식하지만 이를 그대로 이용하였더니 t 를 c 로 인식하거나 i 를 .(comma)로 인식하는 등 정확도가 떨어졌다. 따라서 정확한 인식을 위해 'j TessBoxEditor'라는 유틸리티를 이용하여 box generator 의 수정작업을 진행하였다. 잘못 인식된 box 영역을 직접 지정해주거나 틀린 글자들을 올바른 글자로 인식할 수 있도록 각 철자를 직접 수정해주어야 한다. 이 방법으로 training 을 진행하면 텍스트 인식의 정확도가 훨씬 높아진다.

<트레이닝 추가 전 데이터>	<트레이닝 추가 후 데이터>
Mascsr me basm concepcs and mechodologiss oe di tal signal processing With 1his syscemauc	Master the basic concepts and methodoiogies of di tai signai processing with this systematic
immducum Without me need for an eXcensWe machsmatical backgmund Ths aumors lead me rsader qugh me eundamemal mamemaucal prinprlss undeHymg 1hs opsracion oe I<ey signal processing cschniques. promding simple argumems and cases ramsr than decallsd gensral proofs coverage of pracucal 1mp1emsn1auon7 discussm of 1hs limicacions of msmods7 and plsnueul MATLAB IIIUsUations alloW readers co bsuer connecc msory and pracuce. A eocus on algorithms chat ars of meorsucal impmance or ussnu m real—World appncacions ensures 1ha1 studens cover macerial releVanc co engineering practics and sqmps scudencs and practiuoners alike Wh ms baslc prinprlss nscessary 10 apply DsP cechniques co a Variscy oe applicacions. chapcers mdude Worked eWanesV promems. and compucer experimsn1s7 Helping scudencs 10 absorb me matenal mey haVe	introductiom without the need for an extensive mathematicai background The authors iead the reader through the fundamentai mathematicai principies underiymg the operation of key signai processing techniquesr providing simpie arguments and cases rather than detaiied generai proofs coverage of practicai impiementatiom discussion of the iimitations of methods, and pientifui MATLAB iiiustrations aiiow readers to better connect theory and practice A focus on aigorithms that are of theoreticai importance or usefui m reai-world apphcations ensures that studend cover materiai reievant to engineering practice and equips students and practitioners aiike with the basic principies necessary to appiy DSP techniques to a variety of appiications Chapters mciude worked exampies, probiems and computer experiments, heiping students to absorb the matenai they have

그림 17~18 (왼) Box data 수정 전, (오) Box data 수정 후

위는 training 을 거치기 전, 후에 인식률의 차이를 보이는 텍스트 인식 결과이다. 첫 번째 문장을 보면 training 이전의 데이터는 t 를 c 로, e 를 s 로, th 를 m 으로 인식하는 등 인식률이 60% 이하로 떨어지지만 training 이후의 데이터는 인식률이 90% 이상으로 상승하였다.

2) 글씨 인식 결과에 따른 AR 구현과 여러개의 AR 단계적 송출로 책 내용 흐름 설명

기존의 AR 어플리케이션은 단순히 하나의 그림에 한가지의 AR 을 보여주었다. 하지만 하나의 AR 로 이야기의 내용을 보여주기엔 한계가 있었고, 본 프로젝트는 node 를 업데이트 하여 sceneview 의 child 로 추가하는 방식을 통해 이야기의 흐름을 알리고 프로젝트의 취지에 맞도록 이야기의 흥미를 돋우려 했다.

그림 19~21 버튼을 누를때마다 AR object가 하나씩 추가되는 모습

7. 기존 source code 출처 및 수정 부분

본 프로젝트는 Google 에서 제공한 codelab 의 open source code 를 이용하여 구현하였다. 기존 source code 를 수정한 부분은 다음과 같다.
먼저 어플리케이션의 FirstActivity 라는 Activity class 를 새로 만들고 캡쳐한 이미지의 ROI crop, image 이진화를 거쳐 AR 을 송출하기 이전 글씨 인식을 진행하였다. 그 후 결과값을 Augmented Fragment 에 context 를 통해 DEFAULT_IMAGE_NAME 을 부여해 주었다.

```
private static final String DEFAULT_IMAGE_NAME = ((FirstActivity) FirstActivity.context_first).
BookName +".jpg";

private static final String SAMPLE_IMAGE_DATABASE = "images.imgdb";
```

기존의 AugmentedImageNode 에서 model 의 FirstActivity 의 결과값에 따라 renderable 을 제작하고 AugmentedImageActivity 에 추가한 Button 의 ButNum 숫자가 달라짐에 따라 renderable 을 node 에 setting 하고 Arfragment 를 업데이트 하였다.

```
mazeNode = new Node();
mazeNode.setParent(this);
mazeNode.setRenderable(mazeRenderable0.getNow(null));

switch(((AugmentedImageActivity) AugmentedImageActivity.context_main).butNum){
    case 1:
```

```
            mazeNode.setRenderable(mazeRenderable1.getNow(null));
            mazeNode.setLocalPosition(new Vector3(0.05f,0, 0));
            break;
        case 2:
            mazeNode.setRenderable(mazeRenderable2.getNow(null));
            mazeNode.setLocalPosition(new Vector3(0.03f,0.1f, 0));
            break;
}
```

8. 검토 및 새로운 방향 제안

지금까지 구현한 프로젝트의 한계점들과 개선방향에 대해 제안하고자 한다. 우선 표지에는 제목이 아닌 다른 텍스트도 함께 포함되어 있는 경우가 많다. 예를 들어 작가나 출판사 등은 주로 표지에 명시되어 있다. 그러나 현재 프로그램에서는 표지에 있는 텍스트는 모두 제목으로 인식하여 추출하기 때문에 해당 책이 database 에 있는 경우라고 하더라도 제대로 인식이 이루어지지 않을 수 있다. 이를 방지하기 위해 ROI 를 이용하여 제목만 있는 부분을 crop 했지만 이는 글씨를 인식하려 crop 한 것이 아닌 임의로 ROI 영역을 지정한 것이다. 글씨부분을 자동으로 인식하고 그 중 일정 크기 이상의 글자만을 제목으로 인식하거나 사용자가 제목에 해당하는 이미지 영역을 직접 지정할 수 있도록 하는 등의 기능을 추가할 수 있을 것이다.
또한 Tesseract-OCR 로 텍스트를 추출하는 과정에서 이미지 이진화만을 전처리과정으로 사용했다. 따라서 배경과 텍스트의 차이가 명확하지 않은 경우에는 책 표지의 배경과 텍스트가 같은 색상으로 이진화되어 Tesseract 에서 텍스트를 추출할 수 없다. 또한 표지에 있는 그림이 제목과 겹쳐있는 경우 그림을 텍스트로 인식할 가능성이 높다. 따라서 다양한 경우에 대해 병렬 전처리 과정을 거쳐 텍스트 영역을 정확하게 검출할 수 있어야한다. 그리고 현재 코드에서는 배경 색상보다 텍스트의 색상이 더 어둡다고 가정하여 Tesseract training 을 진행하였다. 그러나 배경 색상보다 텍스트의 색상이 더 밝은 경우도 존재하므로 이에 대한 추가적인 training 도 필요할 것이다.
마지막으로 현재 어플리케이션에서는 하나의 sceneview 에 node 를 추가하여 한 화면에 여러개의 AR 을 동시에 나타냈다. AR 을 교체하기 위해서는 기존의 node child 를 삭제하고 새로운 node child 를 부여해야한다. 하지만 Tracking 이 되는 동안 node 가 clock 마다 초기화되고 다시 rendering 되는 구조이기때문에 기존의 node 를 삭제할 수 없었다. AR 을 교체하는 방식을 사용한다면 더 많은 이야기의 흐름을 나타낼 수 있을 것이다.

9. 참고문헌

[1] 「ARCore Augmented Images」, https://codelabs.developers.google.com/codelabs/augimg-intro/index.html?index=..%2F..index#6, 접속일 2020.06.07.
[2] 「(Android) 광학 문자 인식 라이브러리 Tesseract OCR 의 원리」, https://jinseongsoft.tistory.com/41, 접속일 2020.06.07.
[3] 「차량번호판 인식」, http://egloos.zum.com/eyes33/v/6270884, 접속일 2020.06.07.