

**모바일 응용 소프트웨어 설계**

정보통신공학과 12163291강현지

**<Vuforia Image Targets & OBJ loader>**

제출일 ; 2019년 6월 20일

담당교수님 ; 박 인규 교수님

**TODO**

1. 오브젝트 모델링

- Android장비에서렌더링될수있는적합한모델을찾거나(search) 직접모델링

- Triangular faces 모델

2. 모델링한 오브젝트에 대해 오브젝트의 특성에 맞는 움직임을 직접 조작 할 수 있는 modeling Transform 및 UI 구현 ( button, touch 등의 user interaction )

- Layout 단위에서 버튼 등의 구현 시 res -> layout -> camera\_overlay\*.xml 참조

3. Vuforia SDK의 Targets(marker) 범주를 정하고 해당 Target 에서 오브젝트를 위의 사항을 반영하여 렌더링

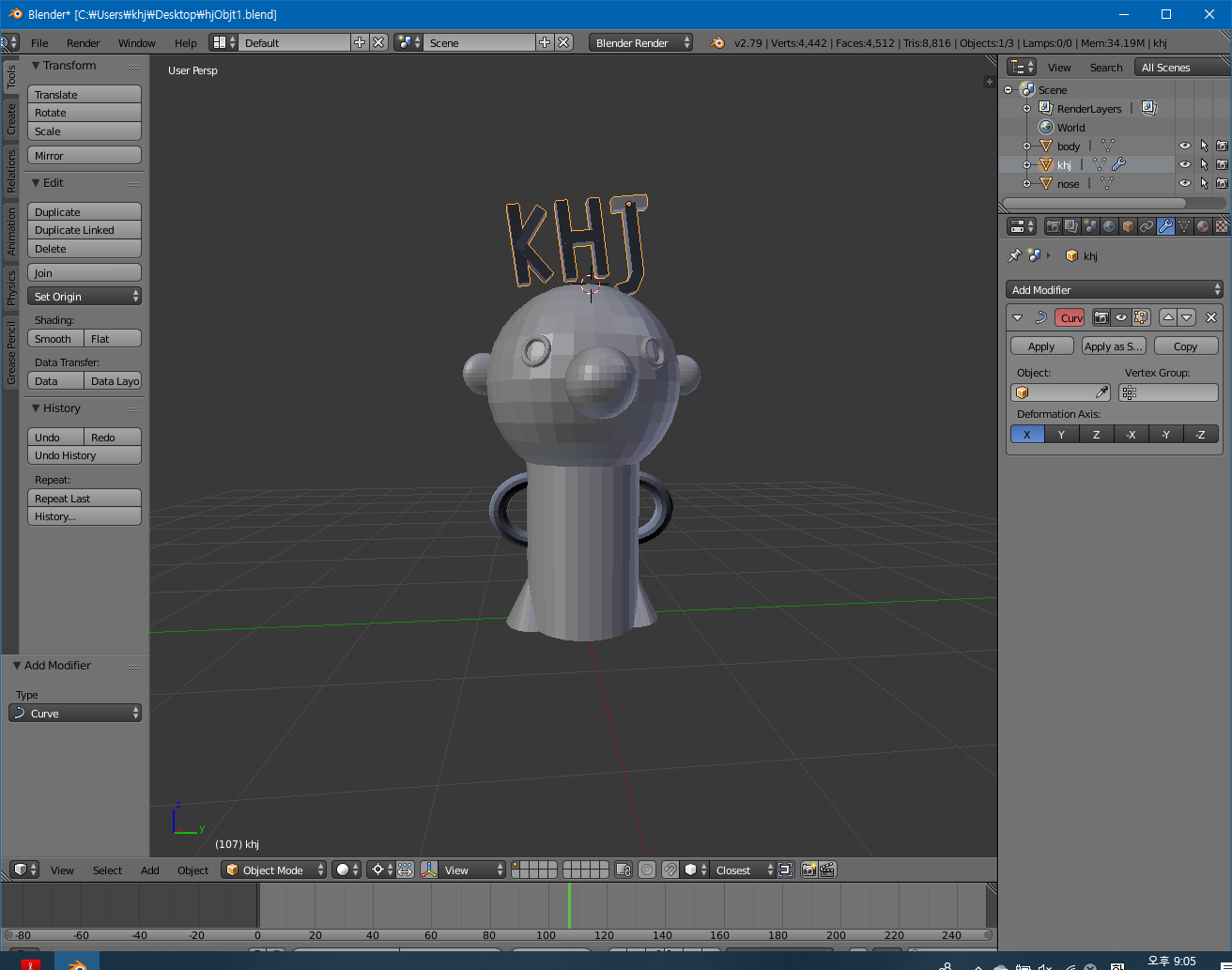
- 수업시간에 배운 Image Targets을 포함한 어느 Vuforia Targets 전부 허용

**1. 오브젝트 모델링**

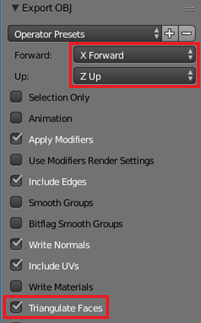
저는 오브젝트를 직접 모델링 하였습니다.

obj는 총 3개로 – 몸, 코, 이니셜 로 이루어져 있습니다.

( 만화 원피스의 버기라는 캐릭터에서 아이디어를 따왔습니다. )



블렌더로 obj 구축 후, Export 해줄 떄의 옵션 체크 사항은 다음과 같았습니다.

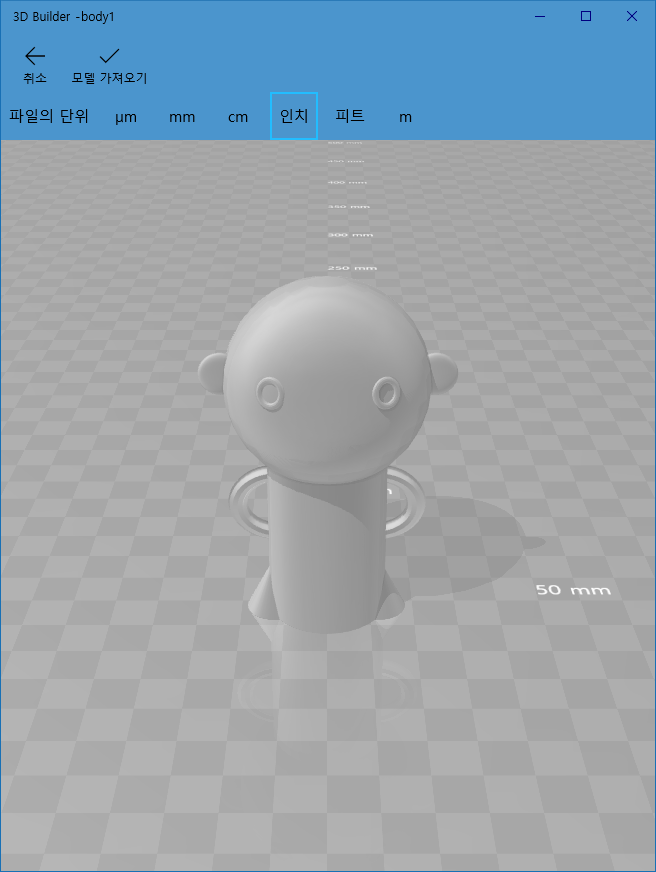


어플리케이션을 실행 했을 때, obj와 눈이 마주친 상황이 default 값이길 원했습니다, 따라서

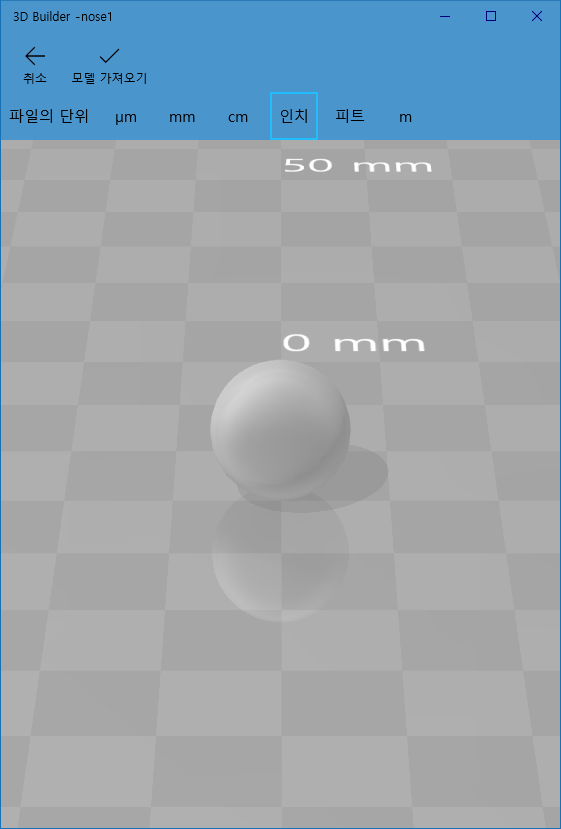
바라보는 방향은 x방향, / UP 방향은 z방향 으로 설정 해 주었고,

Triangular Faces 로 Export 해주었습니다.

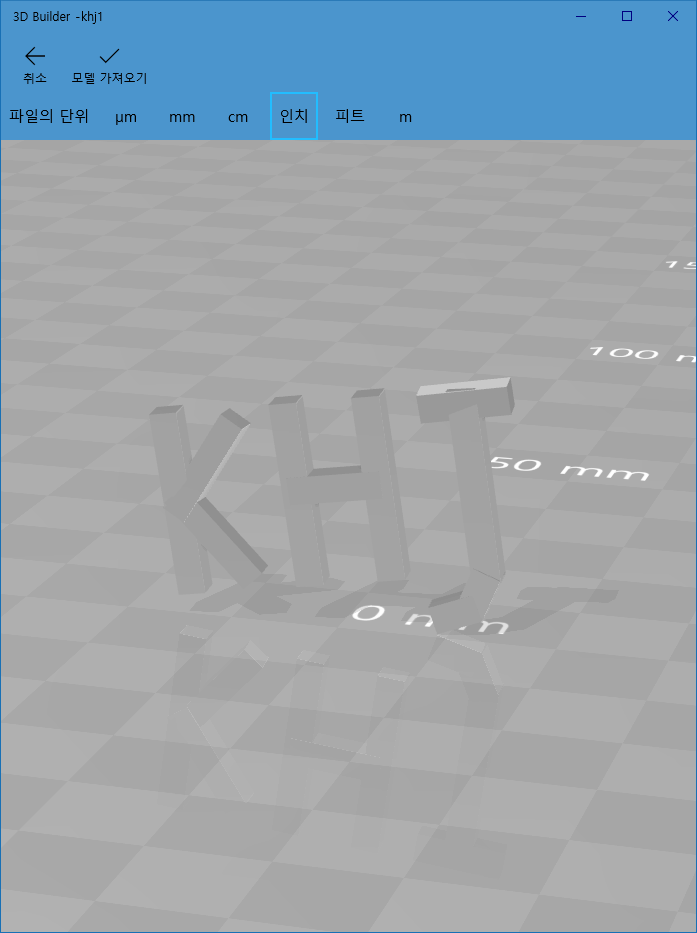
각각의 obj 를 보여드리면 다음과 같습니다.



몸통 obj,



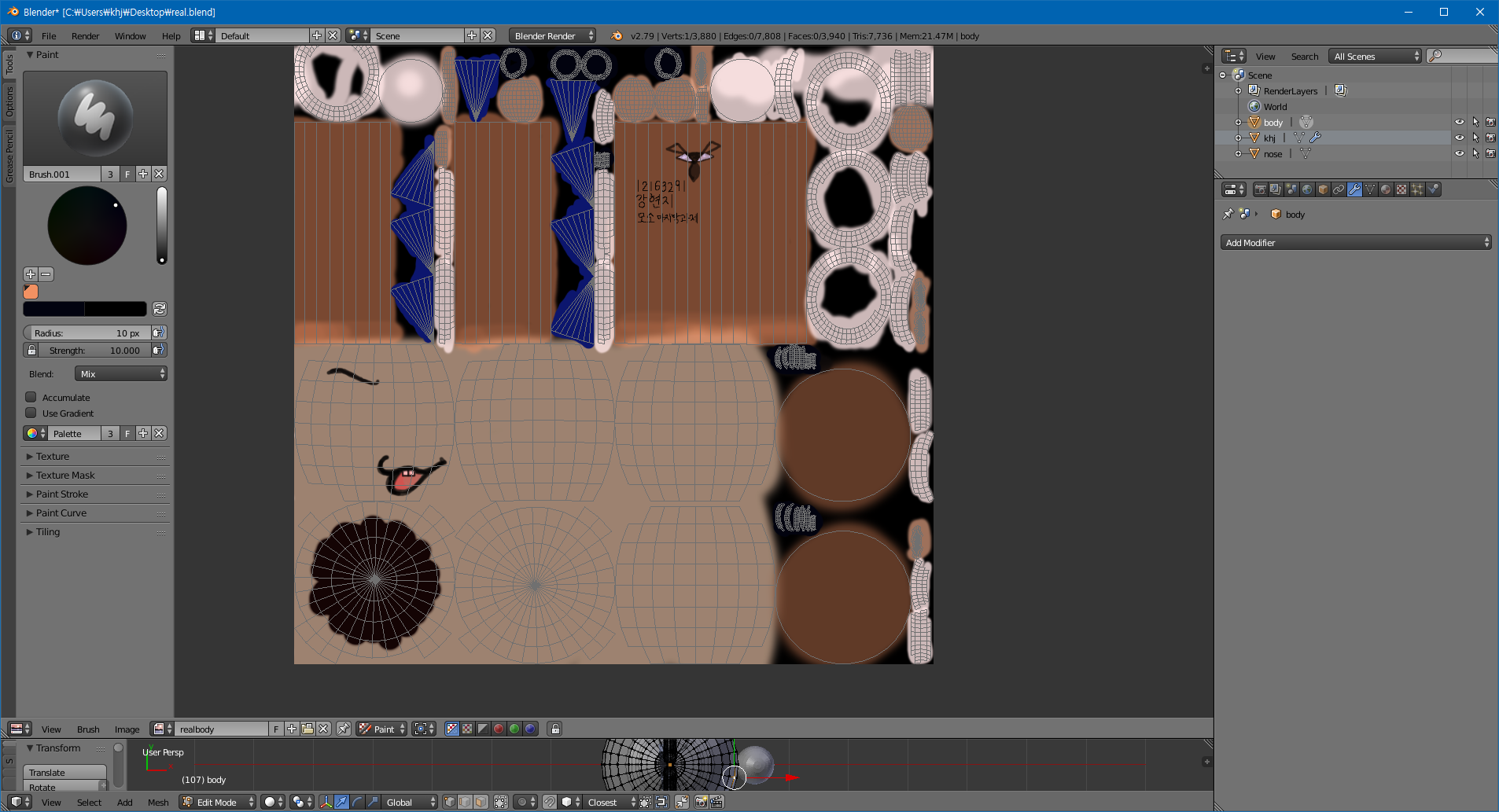
코 obj,



이니셜 obj 입니다.

아래는 obj 대한 텍스처를 직접 그리는 과정을 캡쳐한 사진입니다.

-body obj 대한 Texrure.



코와 이니셜을 따로 움직이고 싶어 3개의 오브젝트를 선택하였고,

저라는 것을 명시하기위해 저의 이니셜 ( KHJ ) 을 구현해 주었습니다.

최종 모델링 결과는 다음과 같습니다



**2. 모델링한 오브젝트 움직임 구성**

제가 구성한 오브젝트의 움직임에 대한 버튼은 크게 총 5가지 입니다.

**1. 앞뒤좌우 움직임**

**2. 점프**

**5. 원점으로 돌아오기 ( RST 버튼 )**

**4. 머리위 이니셜.obj 회전( 이니셜 obj 의 자전)**

**5. 코.obj 펀치! (코 obj 의 앞으로 날라갔다 돌아오기)**

입니다.

움직임에 대해 설명 이전에, **다중 obj 를 구현하기** 위해 작성한 코드에 대한 설명을 하겠습니다.

각 행동을 구현하기 위한 대략적 코드는 다음과 같습니다.

MeshObject model;  
MeshObject model2=null;  
MeshObject model3=null;

MeshObject 변수를 3개 설정해줍니다 . obj Model이 3개 동시 사용되어야 하기 때문입니다.

Matrix.*multiplyMM*(modelMatrix, 0, viewMatrix, 0, modelMatrix, 0);  
Matrix.*multiplyMM*(modelViewProjection, 0, projectionMatrix, 0, modelMatrix, 0);  
GLES20.*glVertexAttribPointer*(vertexHandle, 3, GLES20.*GL\_FLOAT*, false, 0, model2.getVertices());  
GLES20.*glVertexAttribPointer*(textureCoordHandle, 2, GLES20.*GL\_FLOAT*, false, 0, model2.getTexCoords());  
GLES20.*glBindTexture*(GLES20.*GL\_TEXTURE\_2D*, mTextures.get(textureIndex + 1).mTextureID[0]);  
GLES20.*glUniformMatrix4fv*(mvpMatrixHandle, 1, false, modelViewProjection, 0);  
GLES20.*glDrawArrays*(GLES20.*GL\_TRIANGLES*, 0, model2.getNumObjectVertex());

위의 함수를 반복하면 obj 하나를 구현할 수 있습니다.

이를 함수 하나로 설정하여 활용하면 더 효율적인 사용이 가능합니다.

이제, **움직임 구성에 대한 코드 설명을** 하면 다음과 같습니다.

ImageTargetRenderer.java 의,

private void renderModel(float[] projectionMatrix, float[] viewMatrix, float[] modelMatrix, int textureIndex)

renderModel 함수의 내용을 수정함 으로써, obj의 움직임을 변화 시킬 수 있습니다.

if (*objectSel*==2)

만약 ObjSel의 값이 2면, 제가 만든 obj 들이 나타날 수 있도록 하였습니다.

(이는 ImageTarget.java 의 차원에서 해줍니다. )

1. 앞뒤좌우 움직임

Matrix.*translateM*(modelMatrix, 0, 0, -*nose\_y*, 0); //아니 왜 model2,3 같이 움직이냐??

public void IwantMove(int x)  
{  
 switch(x){  
 case 0: //r  
 *step\_x*+=1.5f;  
 *angle*=90;  
 break;  
 case 1: //l  
 *step\_x*-=1.5f;  
 *angle*=270;  
 break;  
 case 2: //b  
 *step\_y*-=1.5f;  
 *angle*=0;  
 break;  
 case 3: //f  
 *step\_y*+=1.5f;  
 *angle*=180;  
 break;

ImageTarget.java 에서, Button이 눌리면 위의 IwantMove() 함수가 실행되도록 하였습니다.

들어온 값에 따라 움직이는 방향와, obj 의 바라보는 방향이 변합니다. 그에따라, 자연스러운 이동을 할 수 있게됩니다.

2. 점프

case 4: // jump  
 *jumpIndex* = 1;  
 *step\_z* = 0.3f;  
 break;

만약 Jump 버튼이눌리면

if(*jumpIndex* == 1){  
 if(*step\_z* < 3.0f) *step\_z* += 0.3f;  
 else *jumpIndex* = 2;  
} else if(*jumpIndex* == 2){  
 if(*step\_z* > 0.0f) *step\_z* -= 0.3f;  
 else {  
 *jumpIndex* = 0;  
 *step\_z* = 0.0f;  
 }  
}

위의 함수가 실행됩니다.

z축으로 올라갔다가 내려오는 것을 구현해 주었습니다.

자연스러운 움직임을 위해 순차적으로 수가 커지다가 작아질 수 있도록 해주었습니다.

이에는 Index 값이 사용되었습니다.

3. 코 펀치!

Matrix.*translateM*(modelMatrix, 0, 0, -*nose\_y*, 0); //아니 왜 model2,3 같이 움직이냐??

코는 y축으로 움직입니다.

if (*noseIndex* == 1) {  
 if (*nose\_y* < 3.0f) *nose\_y* += 0.2f;  
 else *noseIndex* = 2;  
} else if (*noseIndex* == 2) {  
 if (*nose\_y* > 0.0f) *nose\_y* -= 0.2f;  
 else {  
 *noseIndex* = 0;  
 *nose\_y* = 0.0f;  
 }  
}

점프할 때와 유사한 코드를 사용해 자연스러운 코 발사를 구현 해 주었습니다..

4. 이니셜 회전

Matrix.*rotateM*(modelMatrix, 0, *angle\_khj*, 0.0f, 0.0f, 1.0f);

머리 위의 이니셜 obj를 회전시키기 위해 angle 변수를 넣은 rotation 함수를 선언해주었습니다.

if (*rotateIndex* == 1) {  
 if (*angle\_khj* < 359) *angle\_khj* += 10;  
 else *rotateIndex* = 2;  
} else if (*rotateIndex* == 2) {  
 if (*angle\_khj* > 0.0f) *angle\_khj* -= 10;  
 else {  
 *rotateIndex* = 0;  
 *angle\_khj* = 0;  
 }  
}

위와 같이, 점프 할 떄와 유사한 코드를 사용해, 머리위의 이니셜을 회전시켜 주었습니다.

**3. Vuforia SDK의 Targets(마커)의 범주를 정하고 이를 반영하여 렌더링**