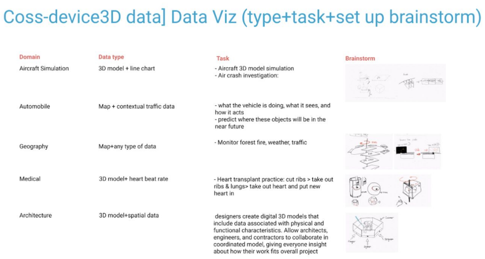
Multi-Surface Handover summary:
Display 3D image on a multi-surface display

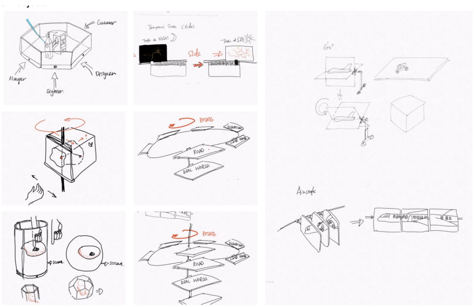
1. History
2. Current framework
3. Current progress + future work

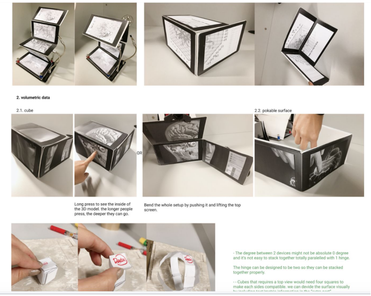
1 HISTORY



Coss-device3D data] Data Viz (type+task+set up brainstorm)

Domain	Data type	Task	Brainstorm
Aircraft Simulation	3D model + line chart	- Aircraft 3D model simulation - Air crash investigation:	
Automobile	Map + contextual traffic data	- what the vehicle is doing, what it sees, and how it acts - predict where these objects will be in the near future	
Geography	Map+any type of data	- Monitor forest fire, weather, traffic	
Medical	3D model+ heart beat rate	- Heart transplant practice: cut ribs > take out ribs & lungs> take out heart and put new heart in	
Architecture	3D model+spatial data	designers create digital 3D models that include data associated with physical and functional characteristics. Allow architects, engineers, and contractors to collaborate in coordinated model, giving everyone insight about how their work fits overall project	





2. volumetric data

2.1. cube

2.2. pokable surface

Long press to see the inside of the 3D model. the longer people press, the deeper they can go.

Bend the whole setup by pushing it and lifting the top screen.

- The degree between 2 devices might not be absolute 0 degree and it's not easy to stack together totally paralelled with 1 hinge.

The hinge can be designed to be two so they can be stacked together properly.

- - Cubes that requires a top view would need four squares to make each sides compatible. we can devide the surface visually by including text/metric information in the "extra part"

10/2019

- ➔ Explore interesting setups for multi-surfaces
- ➔ Find data type + tasks

11/2019

- ➔ BrainStorm setups with tasks & data type considered.--> Use hinge as connector
- ➔ Literature review of hinge setup design

12/2019

- ➔ Narrow Scope → display 3D image
- ➔ Paper prototype

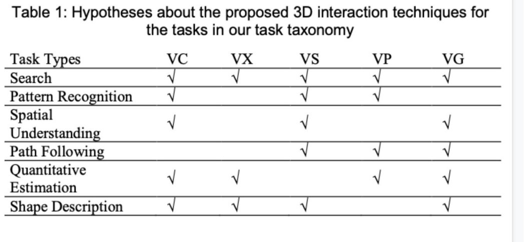
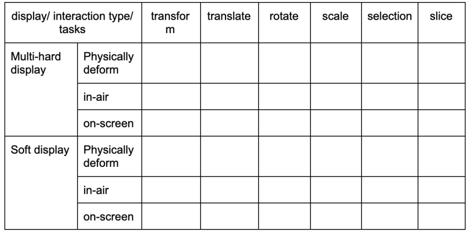
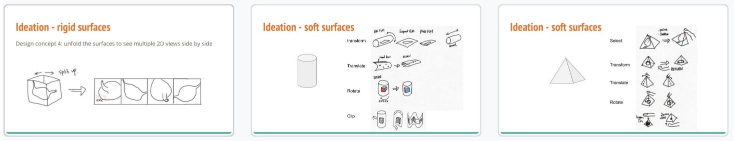


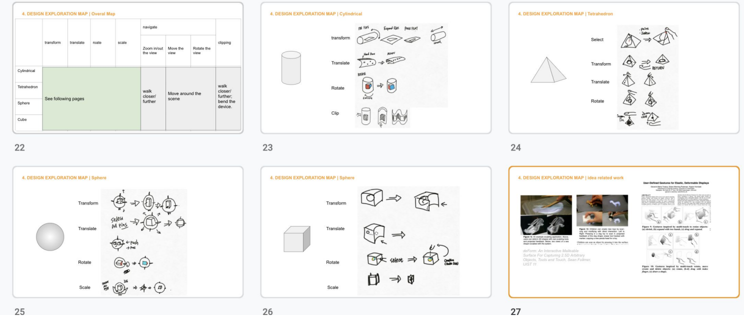
Table 1: Hypotheses about the proposed 3D interaction techniques for the tasks in our task taxonomy

Task Types	VC	VX	VS	VP	VG
Search	√	√	√	√	√
Pattern Recognition	√		√	√	
Spatial Understanding	√		√		√
Path Following			√	√	√
Quantitative Estimation	√	√		√	√
Shape Description	√	√	√		√



display/ interaction type/ tasks		transfor m	translate	rotate	scale	selection	slice
Multi-hard display	Physically deform						
	in-air						
	on-screen						
Soft display	Physically deform						
	in-air						
	on-screen						




4. DESIGN EXPLORATION MAP | Overal Map
navigate
transform
translate
roate
scale
Zoom in/out the view
Move the view
Rotate the view
clipping
Cylindrical
Tetrahedron
Sphere
Cube
See following pages
walk closer/ further
Move around the scene
walk closer/ further; bend the device.
22
4. DESIGN EXPLORATION MAP | Cylindrical
transform
Translate
Rotate
Clip
23
4. DESIGN EXPLORATION MAP | Tetrahedron
Select
Transform
Translate
Rotate
24
4. DESIGN EXPLORATION MAP | Sphere
Transform
Translate
Rotate
Scale
25
4. DESIGN EXPLORATION MAP | Sphere
Transform
Translate
Rotate
Scale
26
4. DESIGN EXPLORATION MAP | idea related work
User-Defined Gestures for Elastic, Deformable Displays
deForm: An Interactive Malleable Surface For Capturing 2.5D Arbitrary Objects, Tools and Touch, Sean Follmer, UIST 11
27

01/2020

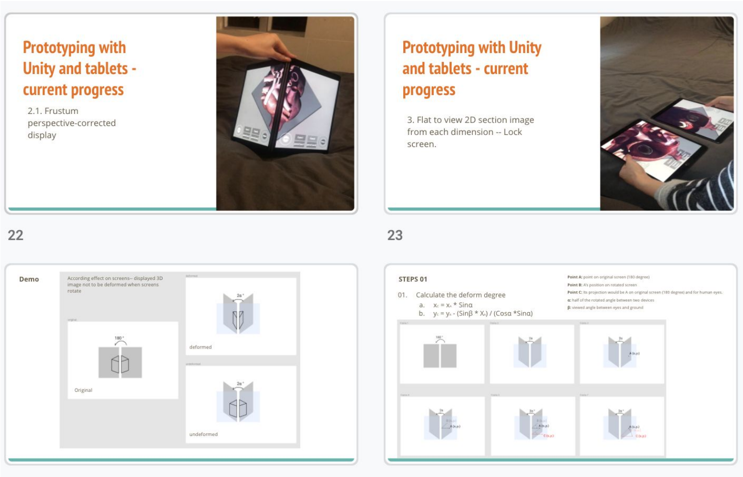
- ➔ Explore specific interaction tasks for volumetric data display for the elicitation study
- ➔ Literature research → bimanual + volumetric data display

02/2020

- ➔ Technical spec design
- ➔ Elicitation study plan design

03/2020

- ➔ Soft display research space exploration → shape-changing interface research space
- ➔ Specify the four concepts



Prototyping with Unity and tablets - current progress

2.1. Frustum perspective-corrected display

Prototyping with Unity and tablets - current progress

3. Flat to view 2D section image from each dimension -- Lock screen.

22

23

Demo

According effect on screens-- displayed 3D image not to be deformed when screens rotate

STEPS 01

01. Calculate the deform degree
    a. $x_C = x_A * \mathrm{Sin}\alpha$
    b. $y_C = y_A - (\mathrm{Sin}\beta * X_A) / (\mathrm{Cos}\alpha * \mathrm{Sin}\alpha)$

Point A: point on original screen (180 degree)

Point B: A's position on rotated screen

Point C: Its projection would be A on original screen (180 degree) and for human eyes.

α: half of the rotated angle between two devices

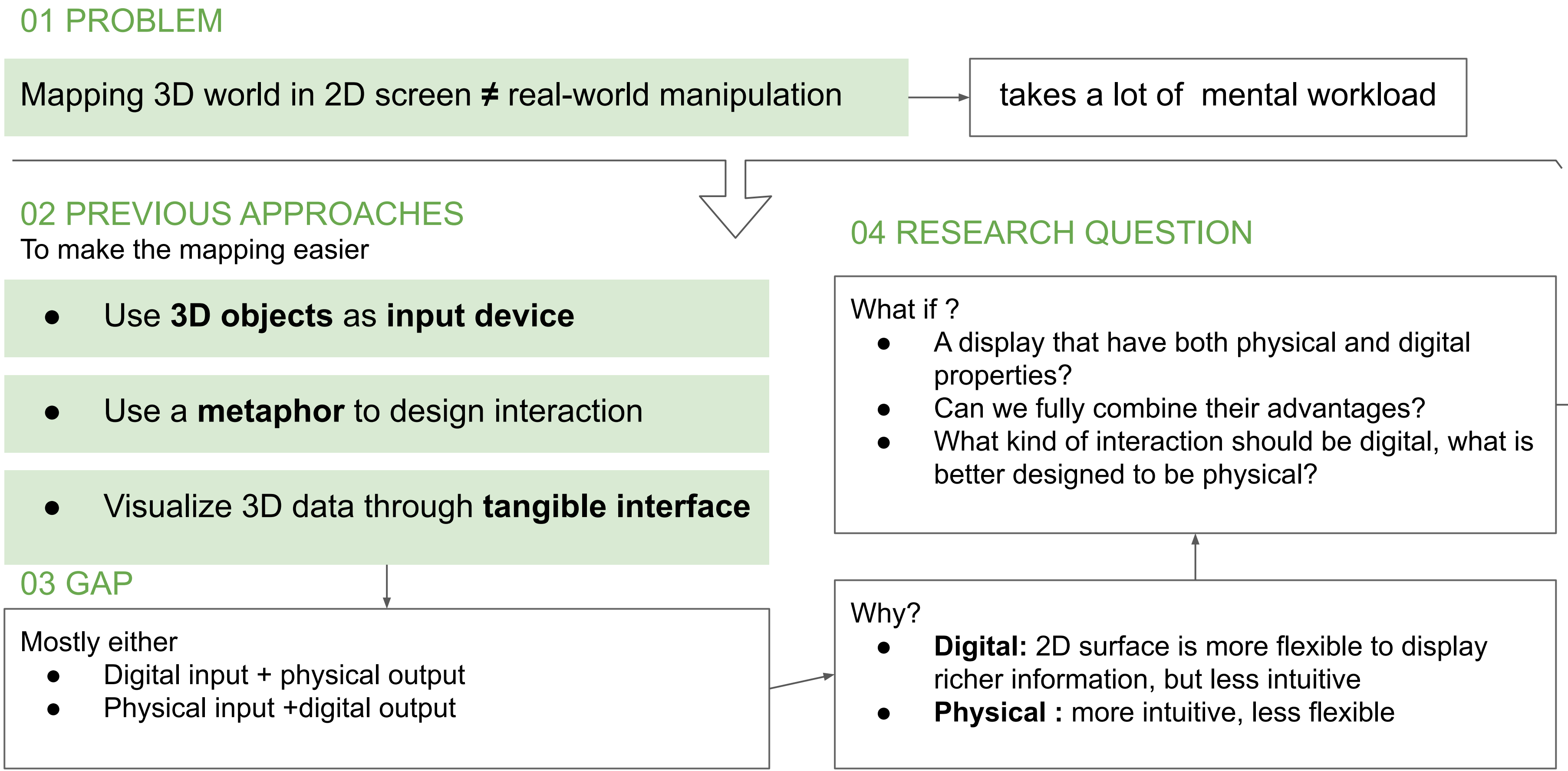
β: viewed angle between eyes and ground

04/2020

- ➔ Prototyping 4 concepts
- ➔ Perspective correction
- ➔ Write the research framework
- ➔ Connector design

2 CURRENT FRAMEWORK

INTRODUCTION



01 PROBLEM

Mapping 3D world in 2D screen ≠ real-world manipulation → takes a lot of mental workload

02 PREVIOUS APPROACHES

To make the mapping easier

- Use **3D objects** as **input device**
- Use a **metaphor** to design interaction
- Visualize 3D data through **tangible interface**

03 GAP

Mostly either

- Digital input + physical output
- Physical input +digital output

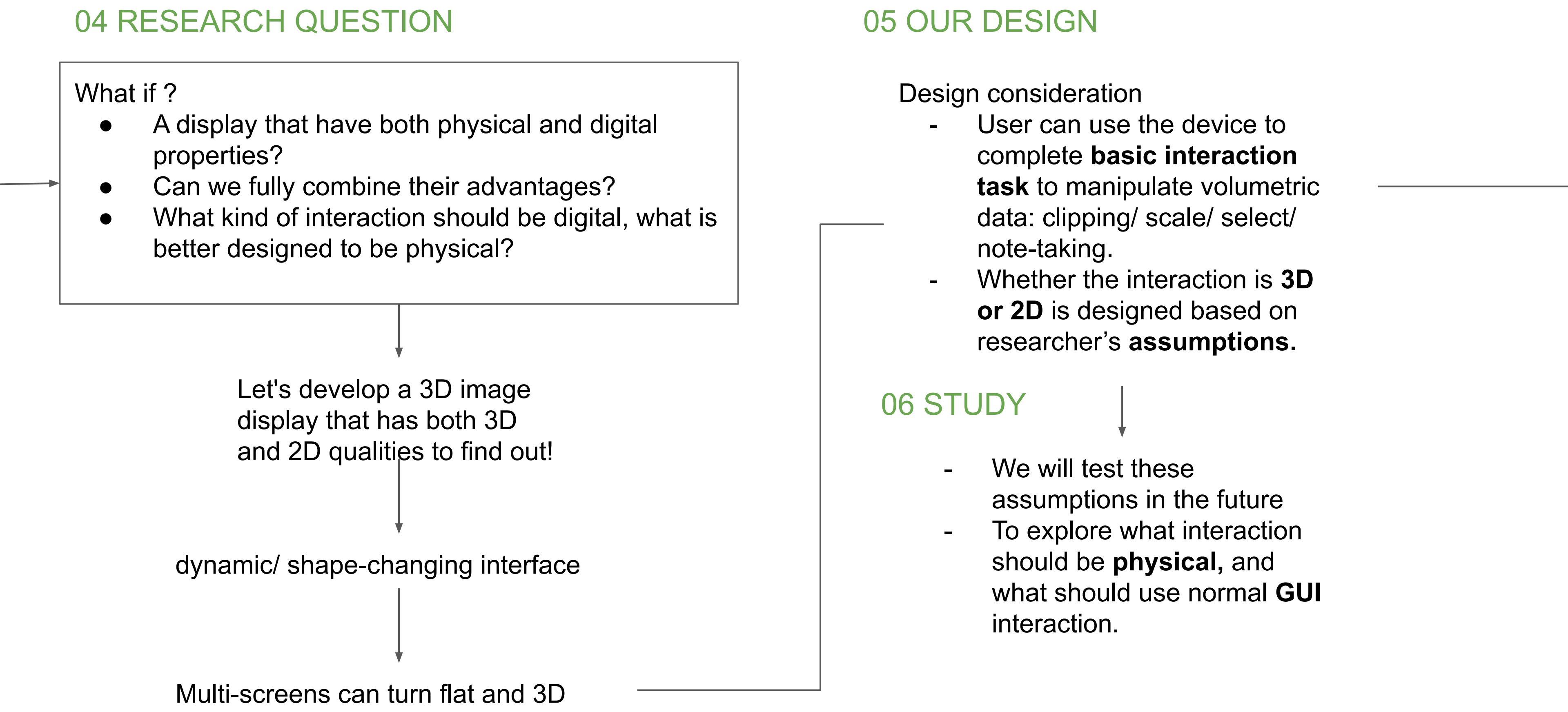
04 RESEARCH QUESTION

Why?

- **Digital:** 2D surface is more flexible to display richer information, but less intuitive
- **Physical :** more intuitive, less flexible

What if ?

- A display that have both physical and digital properties?
- Can we fully combine their advantages?
- What kind of interaction should be digital, what is better designed to be physical?



04 RESEARCH QUESTION

What if ?

- A display that have both physical and digital properties?
- Can we fully combine their advantages?
- What kind of interaction should be digital, what is better designed to be physical?

Let's develop a 3D image display that has both 3D and 2D qualities to find out!

dynamic/ shape-changing interface

Multi-screens can turn flat and 3D

05 OUR DESIGN

Design consideration

- User can use the device to complete **basic interaction task** to manipulate volumetric data: clipping/ scale/ select/ note-taking.
- Whether the interaction is **3D or 2D** is designed based on researcher's **assumptions.**

06 STUDY

- We will test these assumptions in the future
- To explore what interaction should be **physical,** and what should use normal **GUI** interaction.

05 OUR DESIGN

Design consideration

- User can use the device to complete **basic interaction task** to manipulate volumetric data: clipping/ scale/ select/ note-taking.
- Whether the interaction is **3D or 2D** is designed based on researcher's **assumptions.**

06 STUDY

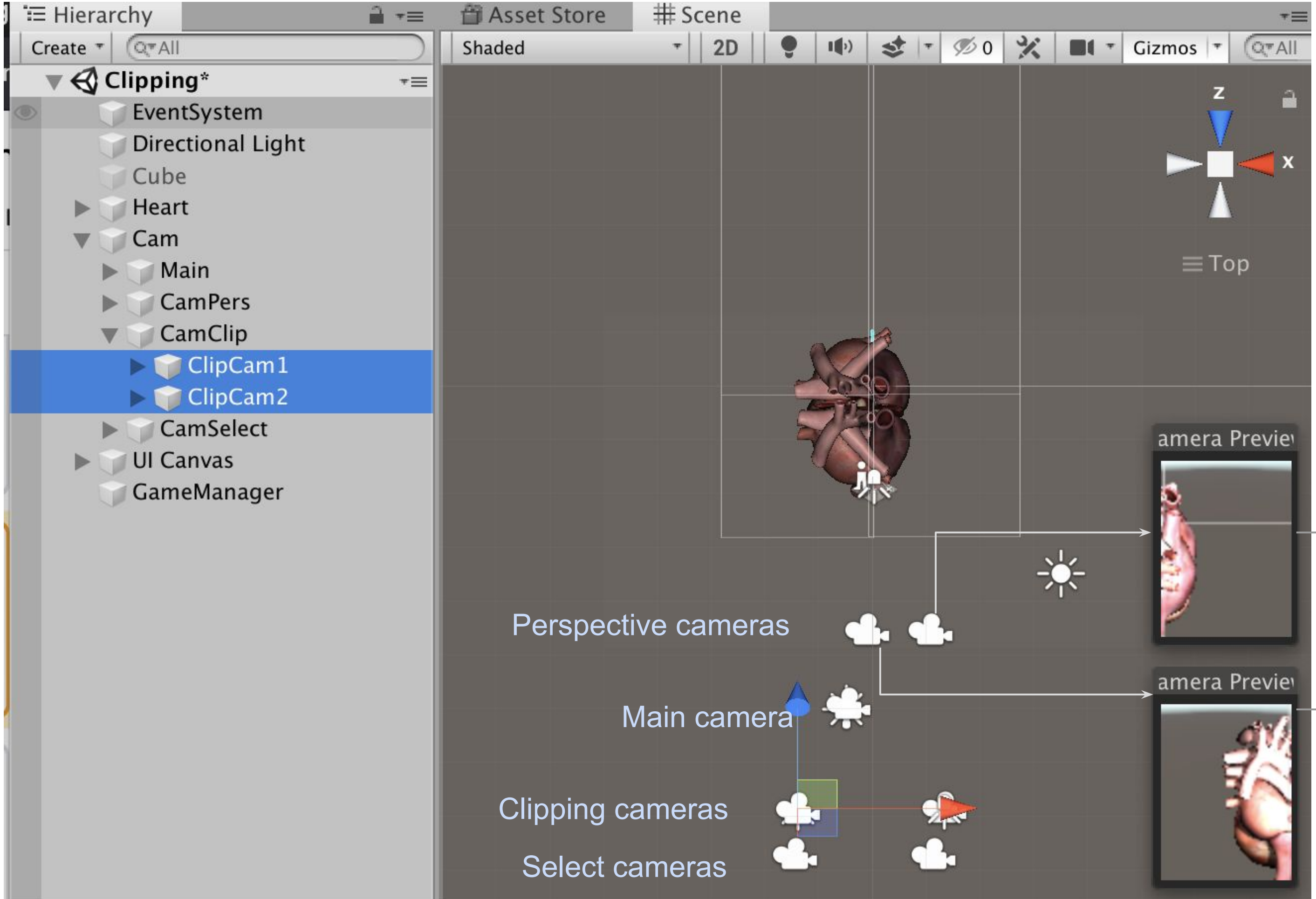
- We will test these assumptions in the future
- To explore what interaction should be **physical,** and what should use normal **GUI** interaction.

07 CONTRIBUTION

1. **Design reference** for displaying 3D data on a dynamic multi-surface display
2. **Study result** of what information should be **displayed** in 3D, what should be displayed in 2D properties. What **interactions** should be displayed in 3D or 2D.

3 CURRENT PROGRESS + FUTURE WORK

STRUCTURE > left/right view



Two cameras:

Cameras for perspective correction

Left camera → simulate the view of the screen on the left

Right camera → simulate the view of the screen on the right

Same for Clipping cameras and select cameras

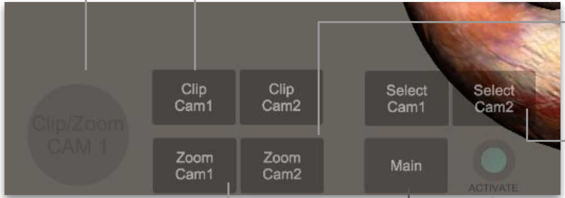
Main camera - default view

STRUCTURE > functionality simulation

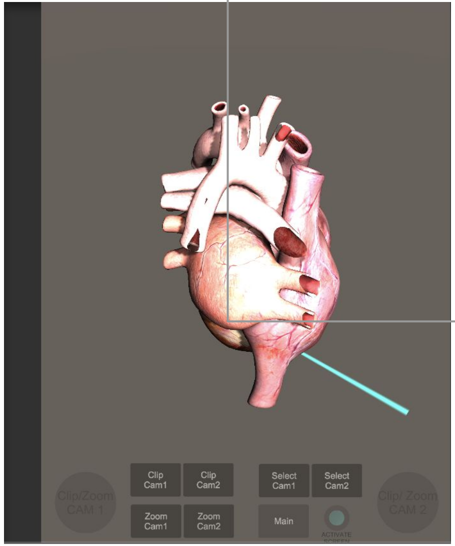
Long press
the button
to clip

Hit to select camera

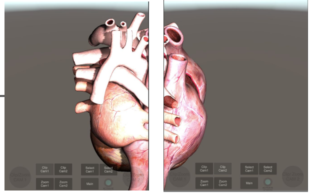
Hit to select camera



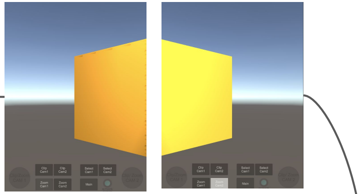
Hit to
select
camera



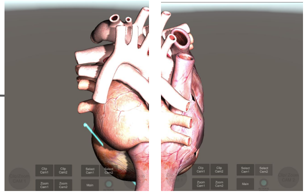
01 CLIPPING



02 ZOOM



03 SELECT



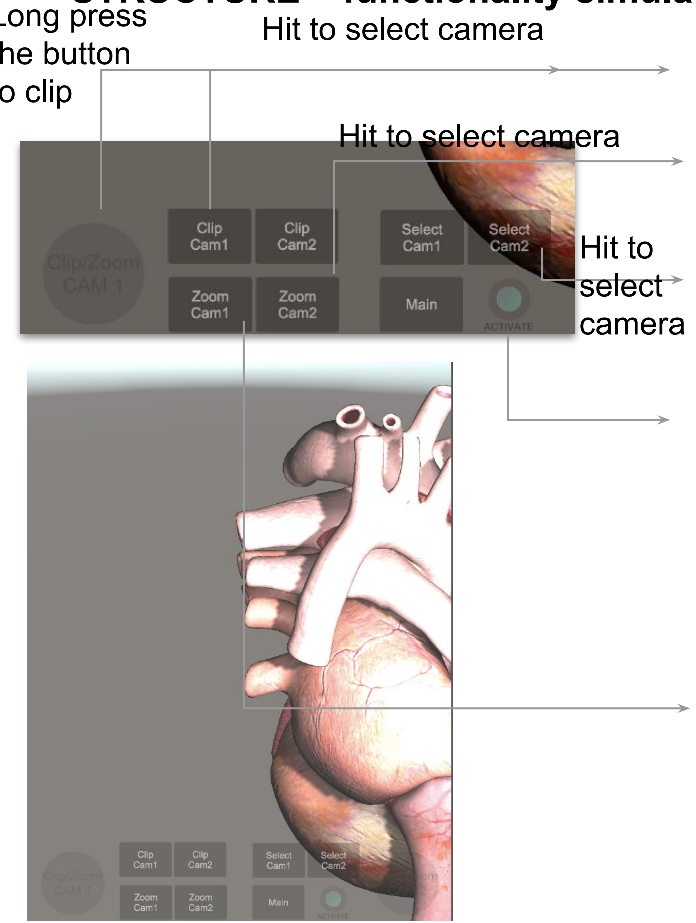
04 EXPAND TO
VIEW IN 2D

05 NOTE TAKING

06 Perspective
Correction while
deforming

07 Sync the views on
two devices

STRUCTURE > functionality simulation



Long press
he button
o clip

Hit to select camera

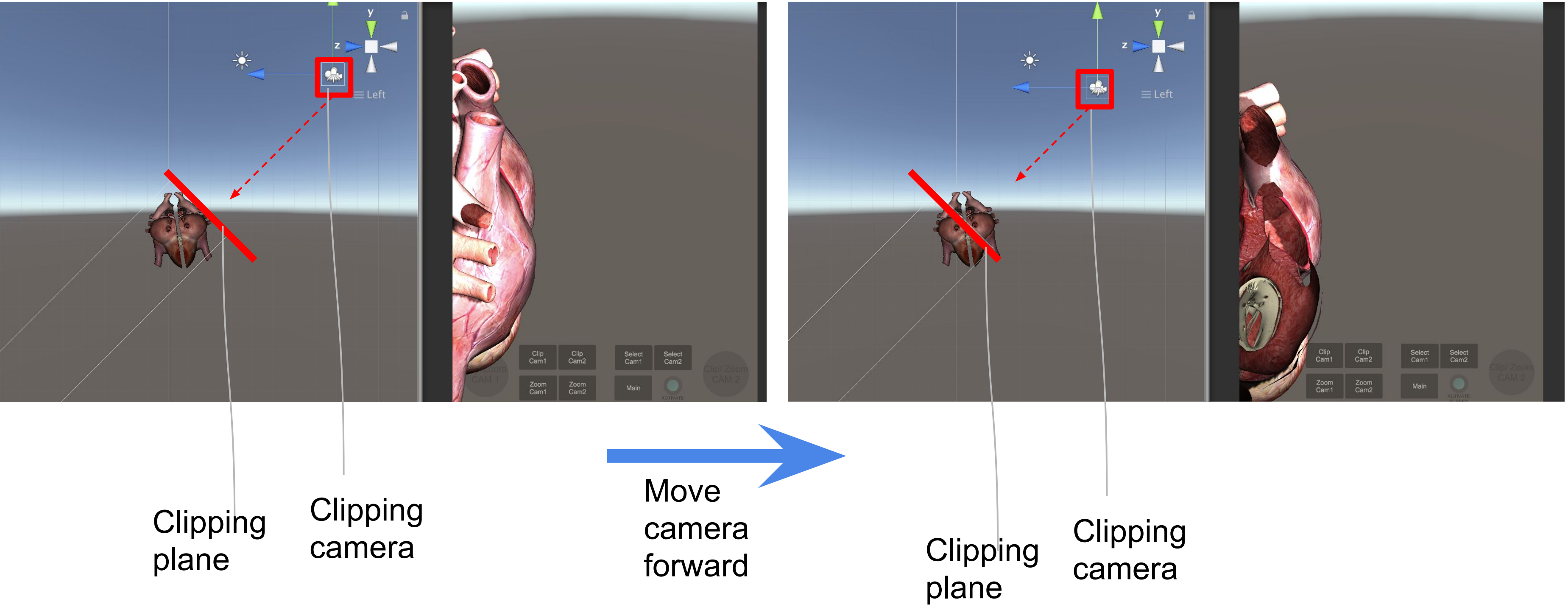
Hit to select camera

Hit to
select
camera

- 01 CLIPPING ➔ Move screens “in” to clip → move cameras forward t
the target object
- 02 ZOOM ➔ Zoom in/ out → move cameras forward/backward to t
target object
- 03 SELECT ➔ Drag the pointing ray to the pointed area (for
now it’s “move the ray to the pointed area)
- 04 EXPAND TO VIEW IN 2D ➔ Freeze the image at current status → turn off gyro
- 05 NOTE TAKING ➔ Sketch based on touch
- 06 Perspective Correction while deforming ➔ Alter camera’s view; also need to combine the
syn function for further simlulation
- 07 Sync the views on two devices ➔ Other image on a device synchronize with the
other. → Unity package: proton (in future)

CONCEPT 1

01 CLIPPING/ ZOOM IN OUT



Clipping plane

Clipping camera

Move camera forward

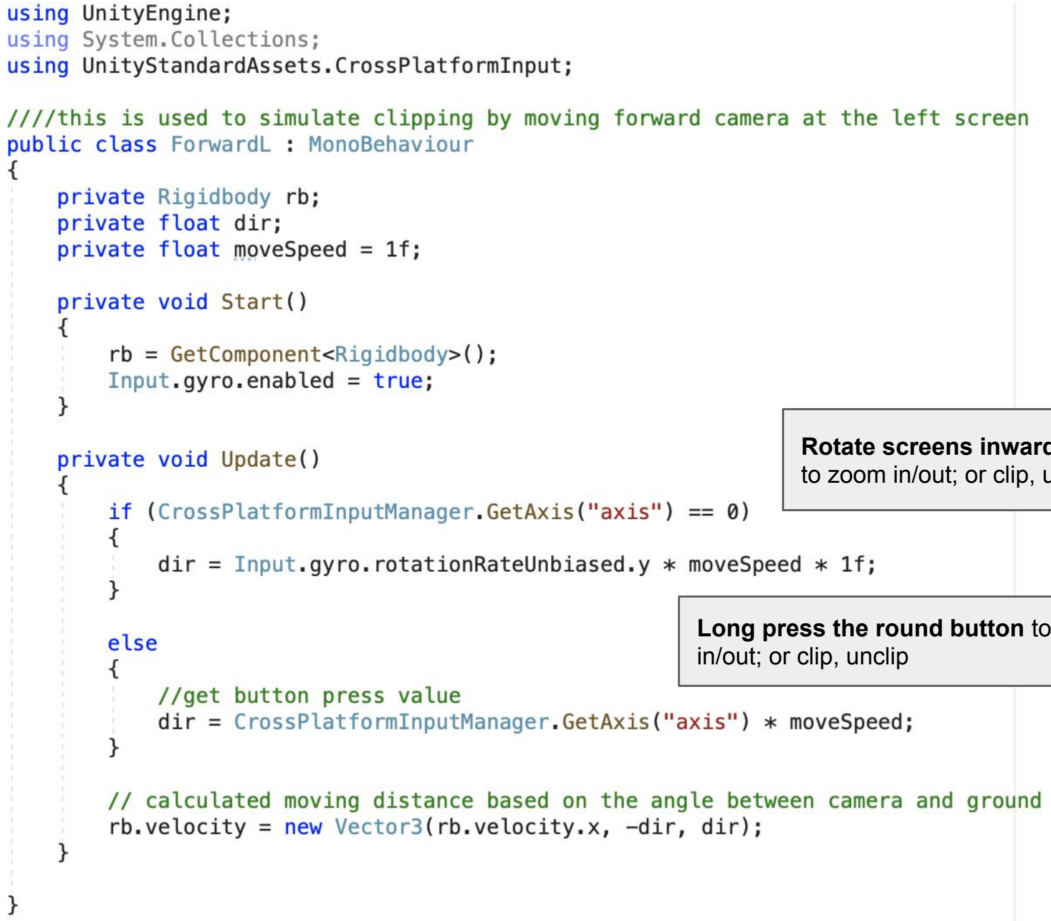
Clipping plane

Clipping camera

CONCEPT 1

01 CLIPPING/ ZOOM IN OUT

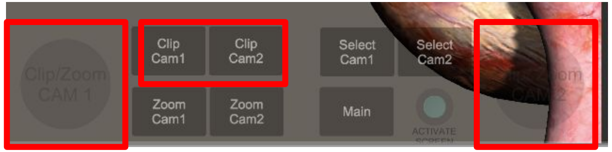
- ➔ Move the cameras forward → image become bigger; the near clipping plane cut in
- ➔ Move the cameras backward → image become smaller; the near become smaller; the near clipping plane cut out

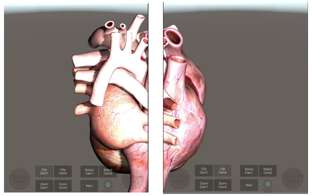


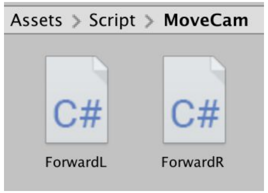
```
using UnityEngine;
using System.Collections;
using UnityStandardAssets.CrossPlatformInput;

////this is used to simulate clipping by moving forward camera at the left screen
public class ForwardL : MonoBehaviour
{
    private Rigidbody rb;
    private float dir;
    private float moveSpeed = 1f;

    private void Start()
    {
        rb = GetComponent<Rigidbody>();
        Input.gyro.enabled = true;
    }

    private void Update()
    {
        if (CrossPlatformInputManager.GetAxis("axis") == 0)
        {
            dir = Input.gyro.rotationRateUnbiased.y * moveSpeed * 1f;
        }

        else
        {
            //get button press value
            dir = CrossPlatformInputManager.GetAxis("axis") * moveSpeed;
        }

        // calculated moving distance based on the angle between camera and ground
        rb.velocity = new Vector3(rb.velocity.x, -dir, dir);
    }

}
```

Rotate screens inward/outward to zoom in/out; or clip, unclip

Long press the round button to zoom in/out; or clip, unclip

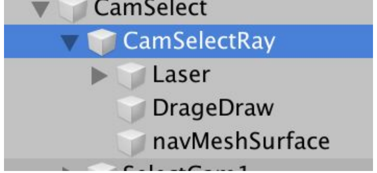




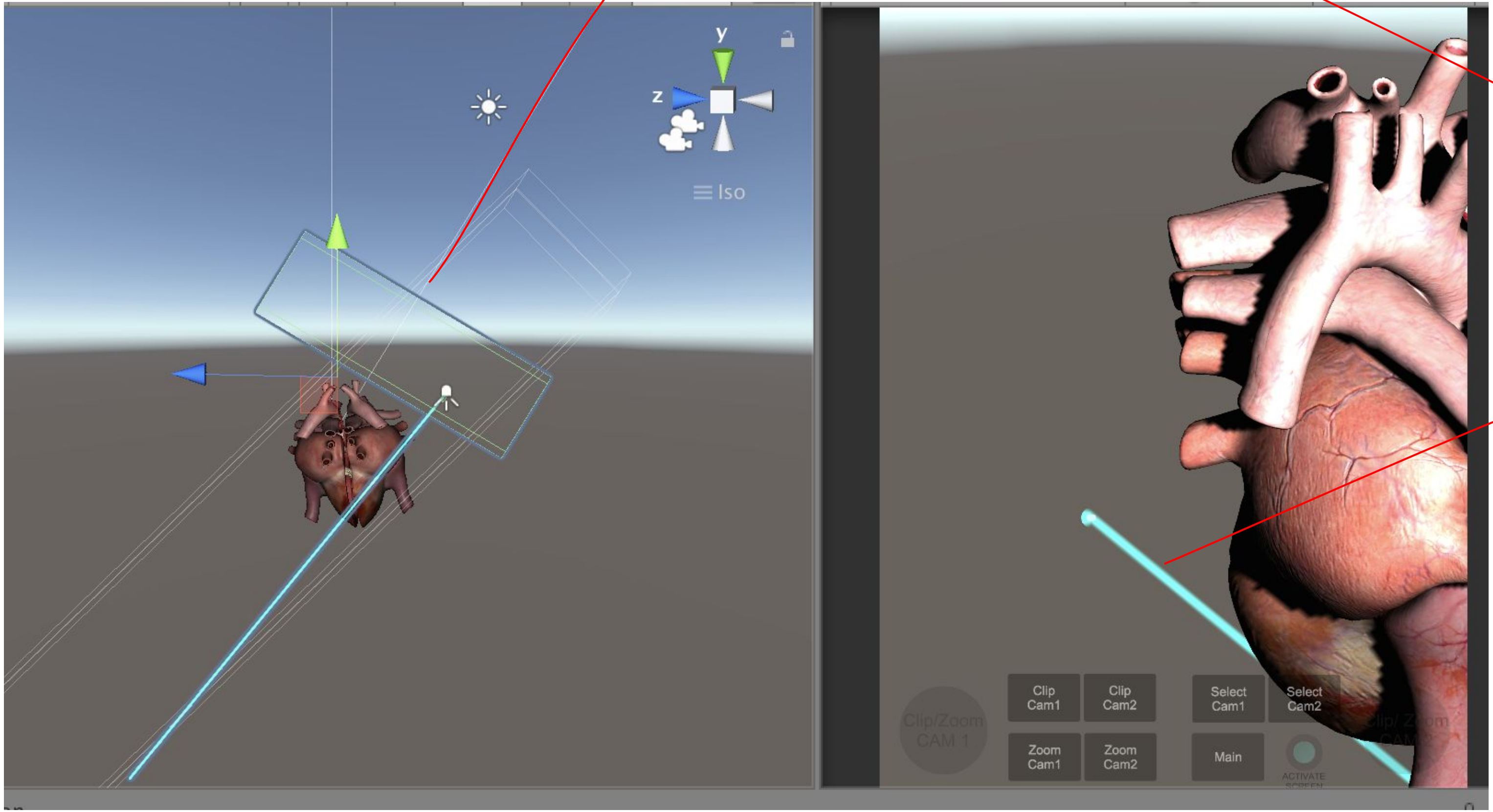


CONCEPT 3

03 SELECT



```
▼ CamSelect
  ▼ CamSelectRay
    ▶ Laser
      DrageDraw
      navMeshSurface
```



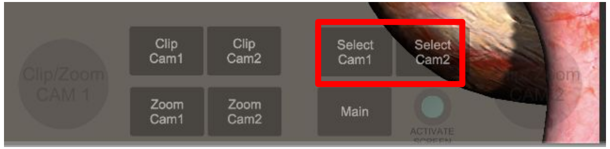
The ray moved on a mesh plane

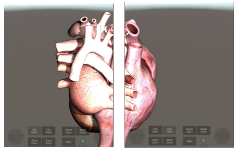
Touch a point → the blue ray follow

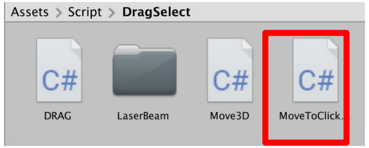
Need to be able to be dragged in the future

CONCEPT 3

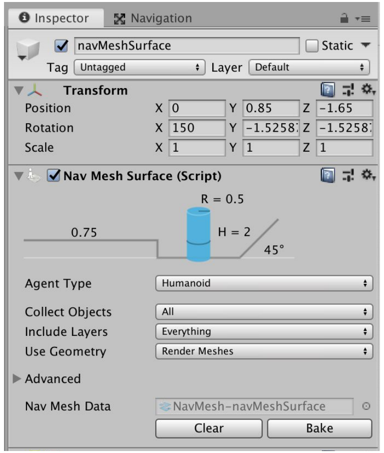
03 SELECT







Render to Navmesh setup



☑ navMeshSurface □ Static

Tag Untagged Layer Default

Transform

	X	Y	Z
Position	0	0.85	-1.65
Rotation	150	-1.52587	-1.52587
Scale	1	1	1

☑ **Nav Mesh Surface (Script)**

Agent Type: Humanoid

Collect Objects: All

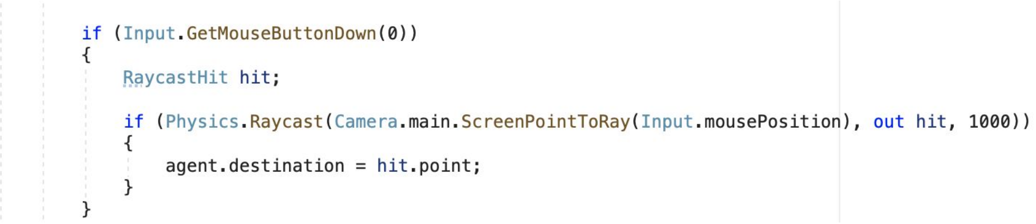
Include Layers: Everything

Use Geometry: Render Meshes

Advanced

Nav Mesh Data: NavMesh-navMeshSurface

Render a surface → make the ray walk on the surface → make it follow touch

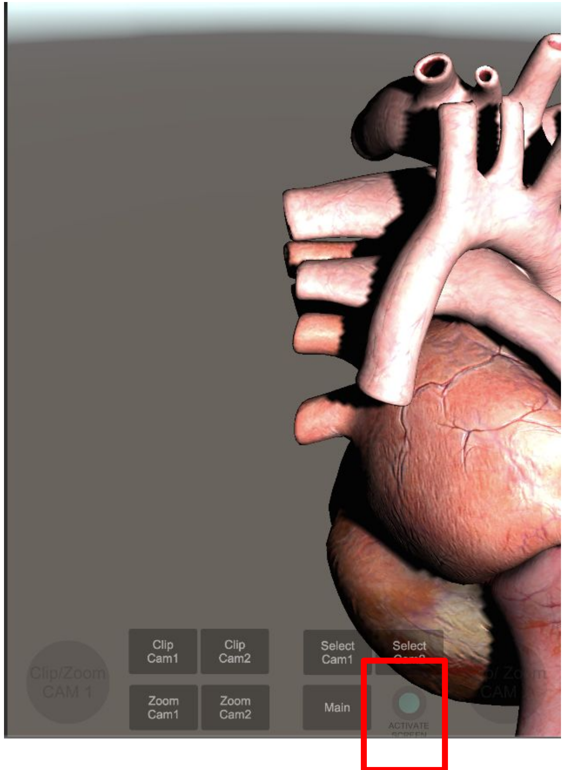


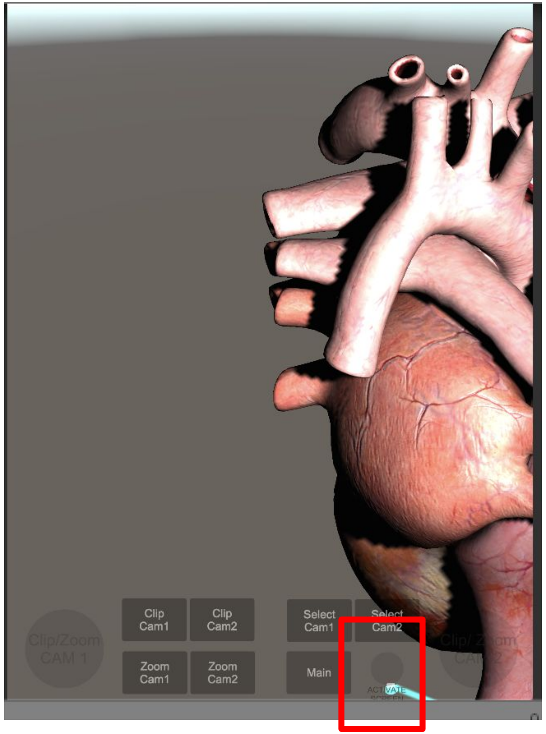
```
if (Input.GetMouseButtonDown(0))
{
    RaycastHit hit;

    if (Physics.Raycast(Camera.main.ScreenPointToRay(Input.mousePosition), out hit, 1000))
    {
        agent.destination = hit.point;
    }
}
```

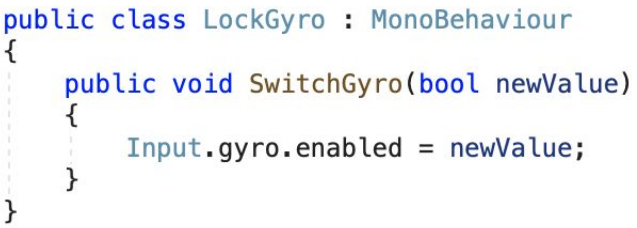
CONCEPT 4

04 EXPAND TO VIEW IN 2D

➔ Freeze the image at current status → turn off gyro





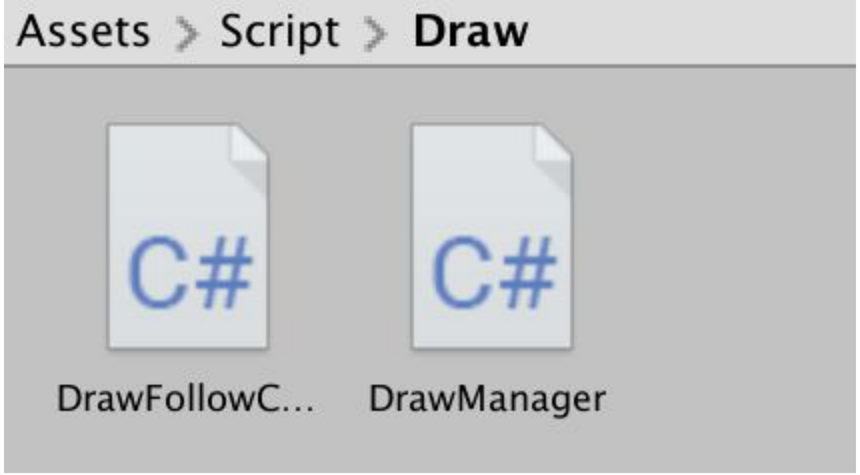


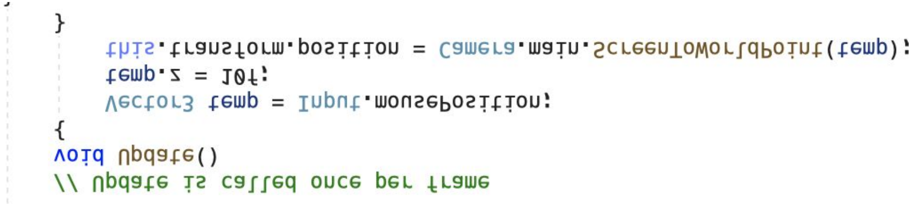
```
public class LockGyro : MonoBehaviour
{
    public void SwitchGyro(bool newValue)
    {
        Input.gyro.enabled = newValue;
    }
}
```

CONCEPT 5

05 NOTE TAKING

➔ Sketch based on touch



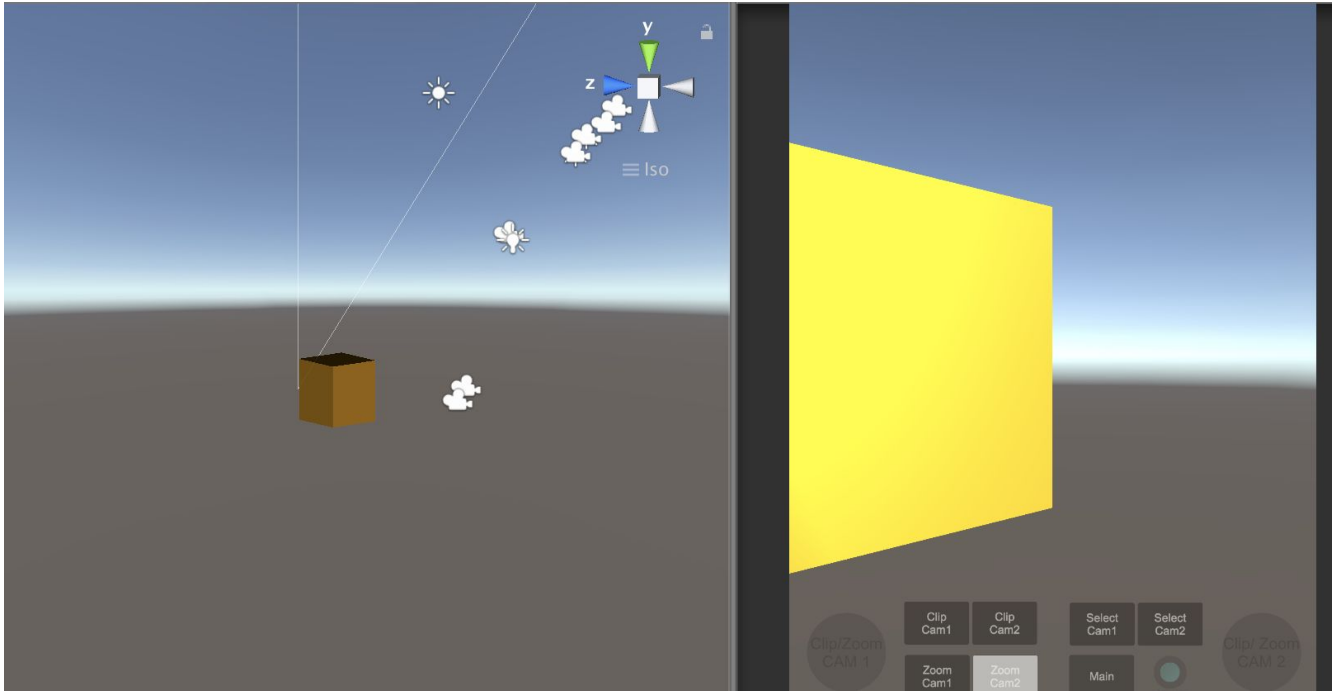


```
// Update is called once per frame
void Update()
{
    Vector3 temp = Input.mousePosition;
    temp.z = 10f;
    this.transform.position = Camera.main.ScreenToWorldPoint(temp);
}
```

CONCEPT 6

06 Perspective Correction while deforming

➔ Alter camera’s view; also need to combine the sync function for further simulation

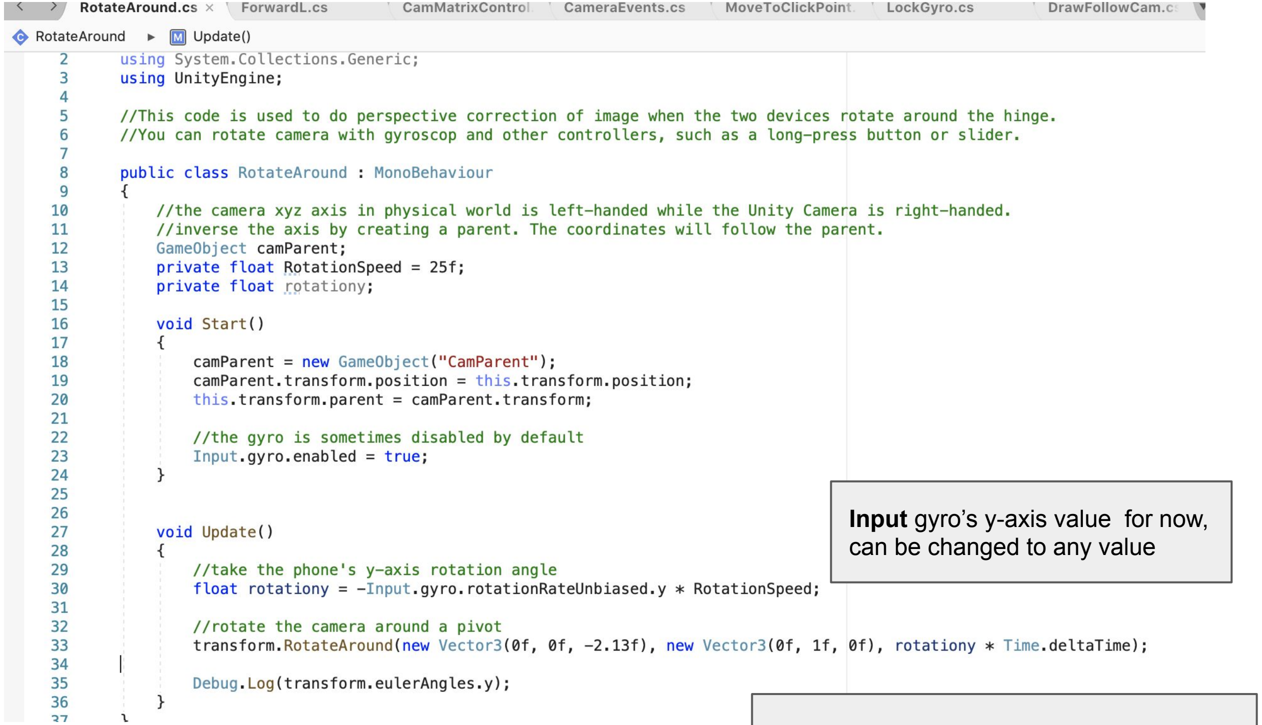


CONCEPT 6

06 Perspective Correction while deforming

➔ Alter camera's view; also need to combine the sync function for further simulation

➔ Rotate the cameras around object to adjust views



```
using System.Collections.Generic;
using UnityEngine;

//This code is used to do perspective correction of image when the two devices rotate around the hinge.
//You can rotate camera with gyroscop and other controllers, such as a long-press button or slider.

public class RotateAround : MonoBehaviour
{
    //the camera xyz axis in physical world is left-handed while the Unity Camera is right-handed.
    //inverse the axis by creating a parent. The coordinates will follow the parent.
    GameObject camParent;
    private float RotationSpeed = 25f;
    private float rotationy;

    void Start()
    {
        camParent = new GameObject("CamParent");
        camParent.transform.position = this.transform.position;
        this.transform.parent = camParent.transform;

        //the gyro is sometimes disabled by default
        Input.gyro.enabled = true;
    }


    void Update()
    {
        //take the phone's y-axis rotation angle
        float rotationy = -Input.gyro.rotationRateUnbiased.y * RotationSpeed;

        //rotate the camera around a pivot
        transform.RotateAround(new Vector3(0f, 0f, -2.13f), new Vector3(0f, 1f, 0f), rotationy * Time.deltaTime);

        Debug.Log(transform.eulerAngles.y);
    }
}
```

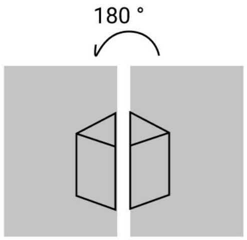
Input gyro's y-axis value for now, can be changed to any value

Output camera's rotation

Demo

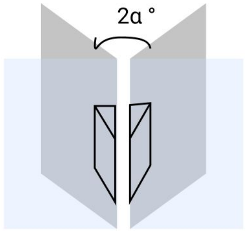
According effect on screens-- displayed 3D image not to be deformed when screens rotate

original



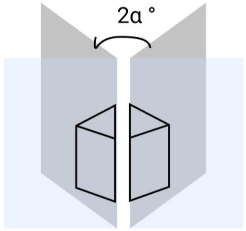
Original

deformed



deformed

undeformed



undeformed

STEPS 01

Point A: point on original screen (180 degree)

Point B: A's position on rotated screen

Point C: Its projection would be A on original screen (180 degree) and for human eyes.

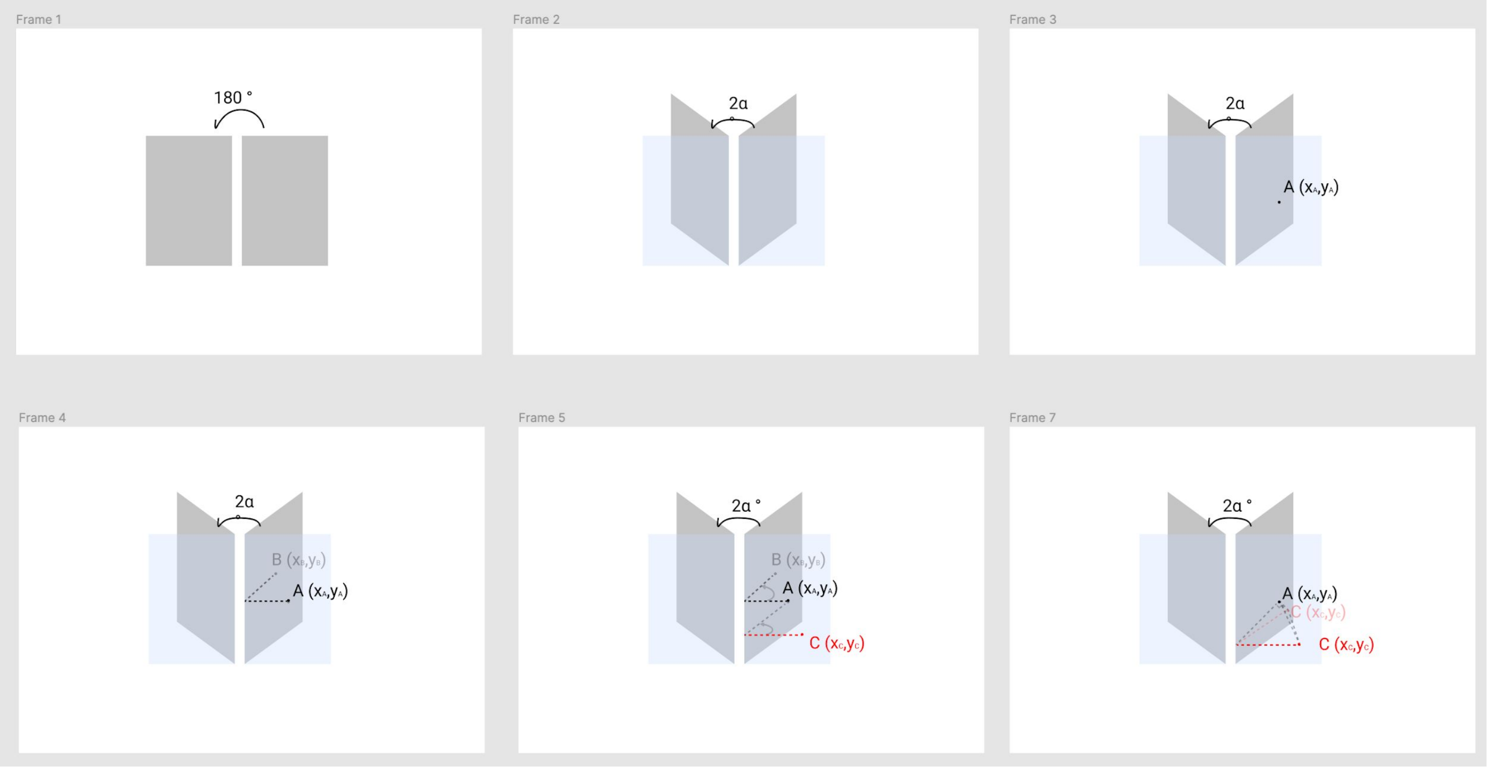
α: half of the rotated angle between two devices

β: viewed angle between eyes and ground

01. Calculate the deform degree

a. $x_C = x_A * \sin\alpha$

b. $y_C = y_A - (\sin\beta * X_A) / (\cos\alpha * \sin\alpha)$

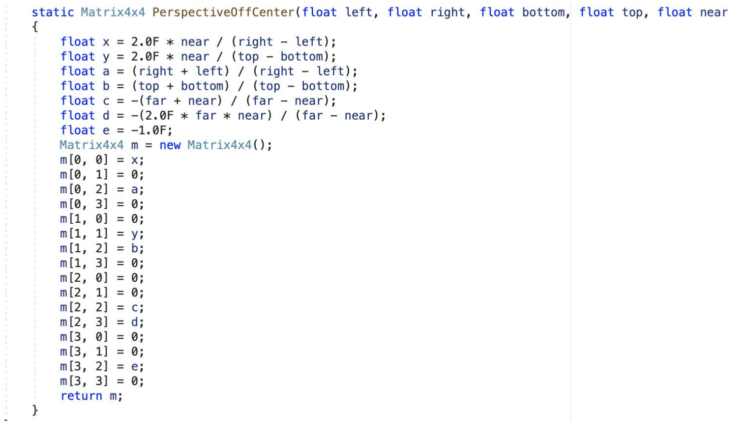


STEPS 02 - a

01. Calculate the deform degree

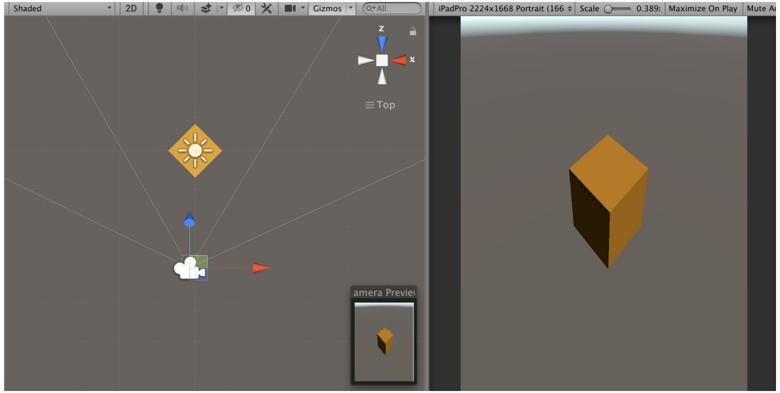
02. Transform the camera view to simulate the unchanging 3D model while rotate.
    a. Move the view center from center to left/right. So the camera transform the image along the hinge instead of the center of the screen.
        i. Camera projection Matrix 4x4 [1]

perspective off-center camera

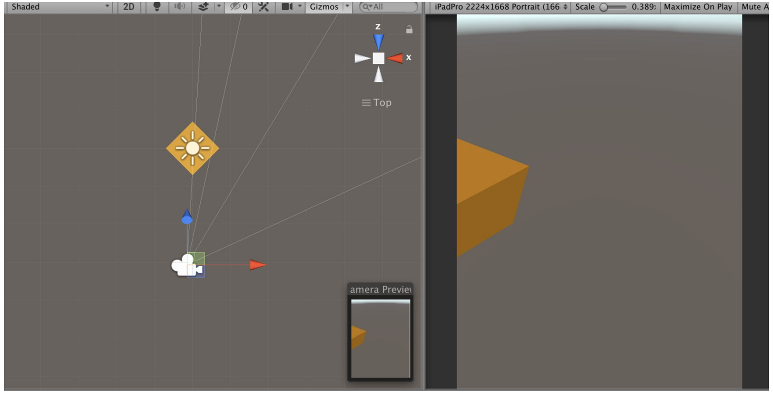


```
static Matrix4x4 PerspectiveOffCenter(float left, float right, float bottom, float top, float near
{
    float x = 2.0F * near / (right - left);
    float y = 2.0F * near / (top - bottom);
    float a = (right + left) / (right - left);
    float b = (top + bottom) / (top - bottom);
    float c = -(far + near) / (far - near);
    float d = -(2.0F * far * near) / (far - near);
    float e = -1.0F;
    Matrix4x4 m = new Matrix4x4();
    m[0, 0] = x;
    m[0, 1] = 0;
    m[0, 2] = a;
    m[0, 3] = 0;
    m[1, 0] = 0;
    m[1, 1] = y;
    m[1, 2] = b;
    m[1, 3] = 0;
    m[2, 0] = 0;
    m[2, 1] = 0;
    m[2, 2] = c;
    m[2, 3] = d;
    m[3, 0] = 0;
    m[3, 1] = 0;
    m[3, 2] = e;
    m[3, 3] = 0;
    return m;
}
```

center camera

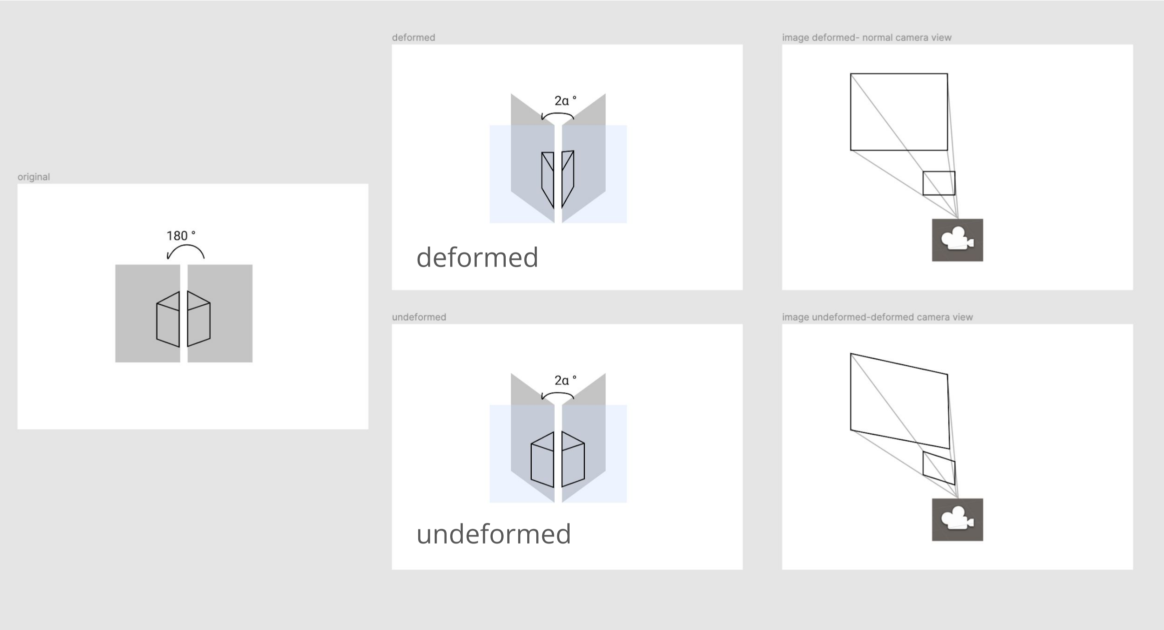


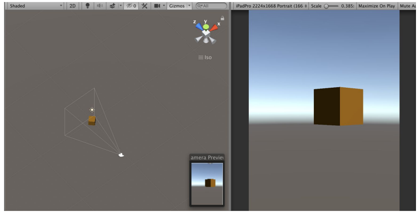
perspective off-center camera

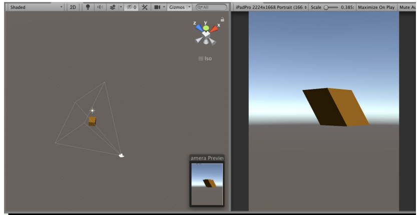


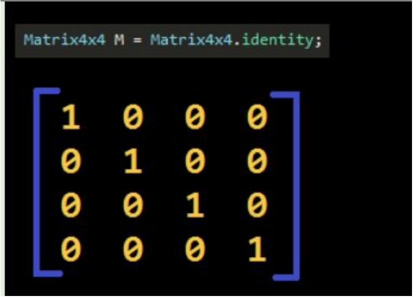
STEPS 02 - b

01. Calculate the deform degree
02. Transform the camera view to simulate the un-deformed 3D image when rotate.
    - a. Move the view center from center to left/right.
    - b. From a standard square to deformed parallelogram-like shape
        1. Camera projection Matrix 4x4 [1]
        2. Camera Matrix Control [3]


original
180 °
deformed
2α °
deformed
image deformed- normal camera view
undeformed
2α °
undeformed
image undeformed-deformed camera view







```
Matrix4x4 M = Matrix4x4.identity;
```

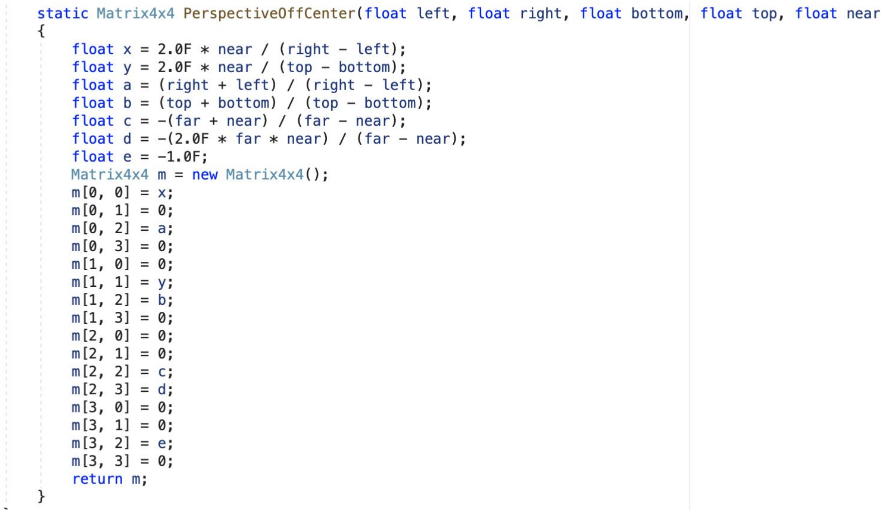
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

CONCEPT 6

06 Perspective Correction while deforming

Future work: alter camera view shape using matri4x4.

perspective off-center camera (refer to unity document)



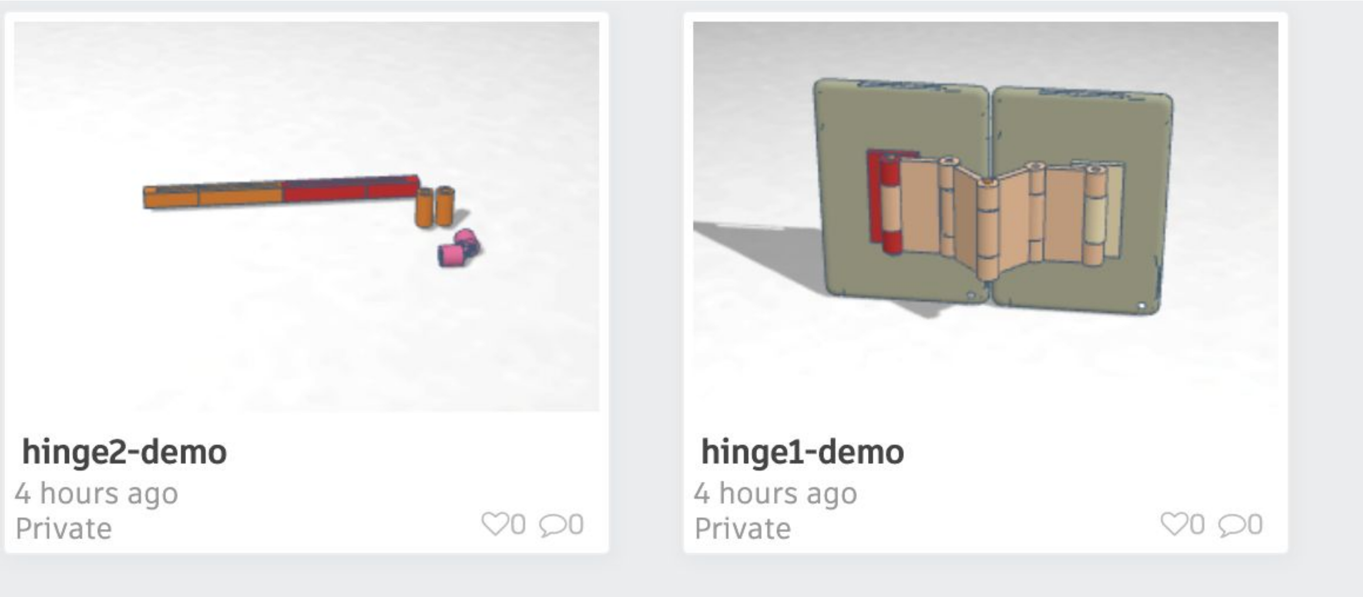
```
static Matrix4x4 PerspectiveOffCenter(float left, float right, float bottom, float top, float near
{
    float x = 2.0F * near / (right - left);
    float y = 2.0F * near / (top - bottom);
    float a = (right + left) / (right - left);
    float b = (top + bottom) / (top - bottom);
    float c = -(far + near) / (far - near);
    float d = -(2.0F * far * near) / (far - near);
    float e = -1.0F;
    Matrix4x4 m = new Matrix4x4();
    m[0, 0] = x;
    m[0, 1] = 0;
    m[0, 2] = a;
    m[0, 3] = 0;
    m[1, 0] = 0;
    m[1, 1] = y;
    m[1, 2] = b;
    m[1, 3] = 0;
    m[2, 0] = 0;
    m[2, 1] = 0;
    m[2, 2] = c;
    m[2, 3] = d;
    m[3, 0] = 0;
    m[3, 1] = 0;
    m[3, 2] = e;
    m[3, 3] = 0;
    return m;
}
```

CONCEPT 7

07 Sync interactions on two devices

Tried Unity old library: Network Manager.
It'll be obsolete soon. Can try its new library
Proton or explore better way

HINGE



hinge2-demo
4 hours ago
Private
♡0 💬0

hinge1-demo
4 hours ago
Private
♡0 💬0