Pico Version : Technical development

SCENE Set-Up:

MENU

- Main Menu
- User Menu

XR Rig (for the Meta Pro Installation use the appropriate V DK)

Holds the Camera and processes most inputs.
Holds the Task visual components (Canvas)
Holds the Application that contains ALL the Scripts to app work
Has the Wall used joint with the Visual component (have same position) to detect
Collisions with the raycast coming from the eyeGaze

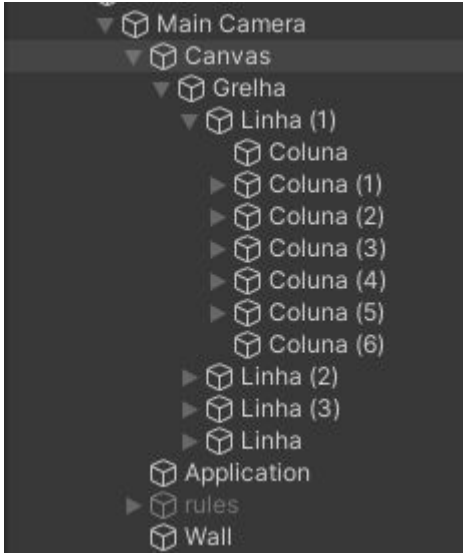
Pico Version : Technical development

“TASK”: Main Components

Canvas : set to be in World Space (allows to follow and pivot according to the camera)

- Grelha holds 4 lines each with 7 columns
- out of 7 Columns, 2 are empty (1st and last) and serve as padding
- 5 inner columbs have a gameObject named after the target/tag name “12” 1st line 2nd inner column (3rd in general)

these images are used by the Application.Prototype Script as the targets. Loading, position getting, and any other operation over these images are processed dynamically by the Prototype script.

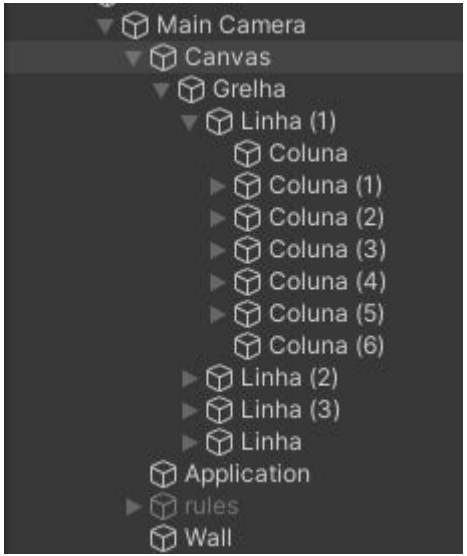


Pico Version : Technical development

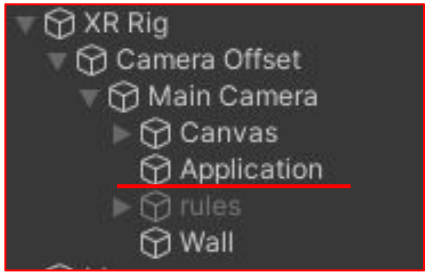
“TASK”: Main Components

Canvas :

Warning!
the 4 x 5(+2 padding) target template can easily be updated. this was what i found to be the one that best covered the full visible/focusable range.



Pico Version : Technical development



“TASK”: Main Components

Application: holds most Scripts, serves as scene Manager.

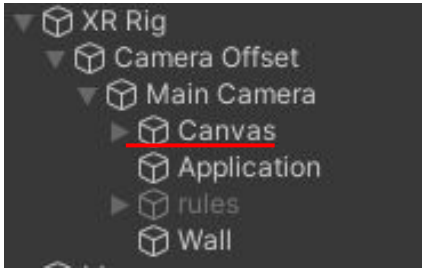
- Prototype
  - Task Loading and set up
  - Settings: (fade in out, showtime, etc durations)
  - Controls in and out of targets + their visual effects
  - Manages the Enters and Leaves of the Task
  - Tag information and current state of task
- Logging
  - Holds : User Id , Task Type Data
  - Creating and Writing to Results.csv
  - gets data (Prototype and GazeRecording) to process Log Template
- Gaze Recording *(uses Tobii.XR)*
  - Controls eyeGaze Ray casting , hits , Screen Coordinate recording
  - TargetOn (debugging tool to see where eyegaze is hitting)
- Pico Click *(uses Pico SDK)*
  - sends RayCast from Camera Center
  - Checks Hit validity
  - highlights “UI” Objects
  - processes “confirm” click

Pico Version : Technical development

“TASK”: Architectural/Design tricks

Having all these components inside of the camera mimics disregard of head movement, rotation ,etc. This allows for all child components to move pivoted by the camera making them look stationary to the user.

Wall is a cube plane extended that is very slightly behind the canvas, this serves to record Hits from the RayCast used in the EyeGazeRay. Since they (hit position and Target Position) are translated into Screen Coordinates this small difference in depth can be disregarded.



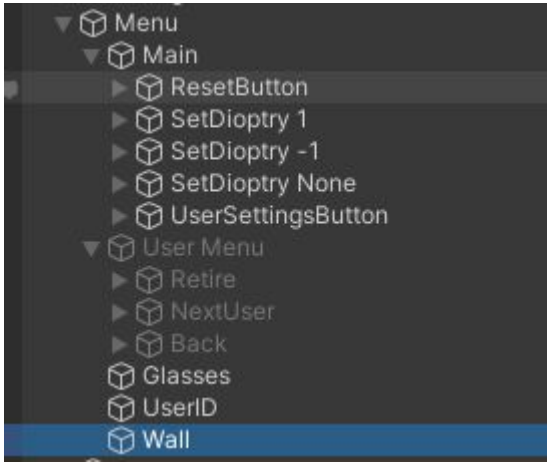
Pico Version : Technical development

“Menu”: Components

Main: has the Buttons from the Main Menu

User Menu: has the Buttons from the UserSettings

These two components hold the Buttons for each page, they function as 2 pages alternating between each other using the:
UserSettingsButton & Back buttons



All buttons are built the same:

- a cube Mesh Renderer
- box collider to be hit by the RayScan
- tag “UI” to be recognized by the PicoClick Script
- Button component, allowing the PicoClick script to trigger the OnClick () event
- Button component to change colour on hover

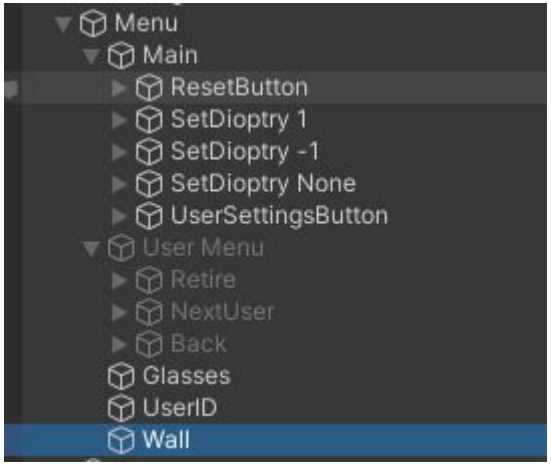
its in the OnClick() that these buttons set the TaskType, change from Main Menu to User Settings,Start the Task, etc.

Pico Version : Technical development

“Menu”: Components

The “Main” component also has a couple of labels to show case some information

- Glasses
- UserId



And a similar trick to the Task where we have a Wall GameObject so that the AIM gameobject (being used as crosshair) doesn't disappear from screen.
This AIM object is moved by a continuous raycast shot from the middle of the Camera, this is controlled by the PicoClick.

Pico Version : Technical development

SCRIPTS : PROTOTYPE

This script is responsible for the “task” in on itself. **As such it is NOT HMD dependant.**

Data :

- List <Images> Dots (array where all Targets will be saved and then reorganized)
- Vector3 tagCoordinates (current Tag coordinates)
- bool on (inside use only, triggers in and out of the task)
- GameObject Grelha (parent to all the Targets)
- float/int used for the timing of the Animations and show time of the Task :
    - fadeInTime
    - showing
    - fadeOutTime
    - timeBetweenDots
    - scaleTime
    - scaleMax
- bool isShowing (internal use flag)
- int activeDot (index of the current Tag in the Randomly Ordered dots List
- string state (shows the current state of the task, if tag is showing, fading in etc)
- float animationTime (internal clock to time the animations)

Pico Version : Technical development

SCRIPTS : PROTOTYPE - Behavior

- On Aplication Start (), get all Images inside the *grelha* component and saves them in *dots* array and resets the Task(randomly ordering the *dots* array and setting all elements to invisible)
- When the Task is turned *ON*, it triggers the *onEnter()* Events , and starts the task.
- During the Task *ON* , using the settings, animates the current target and updates the *state* variable with the current animation state.
- Once the *activeDot* index has reached the end of the *dots* array, it signals the end of this task and triggers the *onEnd*() Event.

OnBegin():
- shuts off the Menu Component .

OnEnd():
- Turns On the Menu Component
- Turns On the PicoClick
- Turns Off the Prototype
- Turns ON the AIM object

Pico Version : Technical development

SCRIPTS : PicoClick

This script is responsible for the Menu UI. This script uses the key “JoystickButton0” which is the name given to the Confirm button on the PICO XR, to use a different SDK simply **change the Key name to what you wish.**

Data :

- TextMeshProUGUI LOG (when you need to see some log during the Menu , connect a label here)
- GameObject target (this is what will serve as a crosshair)
- bool on (inside use only, triggers in and out of the task)
- Color highlightColor (the color of highlight changes)

SCRIPTS : PicoClick - Behaviour

This Script has 2 main behaviours :

- Highlight UI Objects :
  - Every frame a raycast is cast from the center of the Camera
  - when it hits move the target (crosshair) to the hit position
  - if it object hit is Tagged “UI”
  - save the current color , change color to highlight color
  - on leave turn the color back
- Trigger onClick() :
  - if the “`KeyCode.JoystickButton0`” is clicked and the hit is valid call PerformRaycast()
  - if the Object hit is Tagged “UI” trigger its OnClick()

Pico Version : Technical development

SCRIPTS : Logging

This script is responsible to Logging data to the Results.csv. This script doesnt use any specific SDK so it **does NOT need to be changed**
Data :

- Prototype prototype (self loads the prototype)
- GazeRecording gazeRecording (self loads the gazeRecording)
- string taskNumber (holds the data of the taskType)
- int UserId
- string fileName & filePath (used to hold and create a new string)

SCRIPTS : Logging- Behaviour

This Script has many small behaviours :

- CreateCSVFile()
  - If there isn't already a file creates it and creates the Header 1st line.
- Handling UserId :
  - Reads it and Increments it in the *PlayerPrefs*
  - If User gets retired writes a line to the Results.csv *($"{timestamp}, {UserId}, RETIRE, NULL, NULL, NULL, NULL, RETIRE");*
- Loggin :
  - Once the prototype.state is different than "STOPPED"
  - Gets Data from gazeRecording and Prototype
  - Calculates data from hitpoint vs activeTag's position :
    - x and y Deviation
    - distance between both points
  - Writes a Line to the Results.Csv with all this data templated
  - Template: *$"{timestamp}, {UserId}, {taskNumber}, {deviation.x}, {deviation.y}, {distance}, {dotTag}, {state}"*

Pico Version : Technical development

SCRIPTS : GazeRecording

This script is responsible to get Gaze information from the PICO sensors using the Tobii library. **For different installations this SCRIPT MUST BE ALTERED.**
Data :

- GameObject Target & bool targetOn(used for debug, if on target is used as crosshair to mark eyeTracking hits from the Raycast)
- TextMeshProUGUI values_screen & values_hit (another debug tool to show current vector hit coordinates on world and screen)
- Vector3 gazeHit (holds the hit's position)

SCRIPTS : GazeRecording- Behaviour

This Script has 1 single behavior send and record eyeGazeRaycast.

Using the

```
TobiiXR.GetEyeTrackingData(TobiiXR_TrackingSpace.World);
```

we get the information about the gazeRay , knowing its rayOrigin and direction we can recreate a physics raycast .

on a valid hit from the Raycast we save the *WordToSceen* coordinates on the variable ***gazeHit .***

Comments on the Pico Software Version

Although it might seem somewhat complex, the scripts are quite simple and straightforward to understand.

The scene architecture might have a bit more nuance to hit, since it uses the tricks mentioned above .

It all seems to be functional and ready to use. Only lacking in the rest of buttons for all the Standard glasses that will be required.

Unfortunately having in mind the need to change glasses and calibrate after every task some hassle will be added to the testing process.

Advices when Creating other versions

Take special consideration when Building the scene:

- make sure that all Canvas are set to World Space
- the task canvas should be child of the Main Camera (if you want to disregard head Movement)
- Raycast and UI aren’t very compatible when they are sent from the “Virtual” and XR. (reason why i used GameObjects with Meshs and colliders to record my hits)
- Don't make very difficult input options since most of the set up and start of task will have to be done by the participant.
- Use Tags to differentiate objects Hit
- Most Important : the HMD will warp vision and as such some modifications might be needed to the Grelha , test and test until **you can see all the dots** AND they are at **the edge of your gaze amplitude**. The further from the center the more deviations should appear.

Advices when Creating other versions

When it comes to scripts :

- Research well the SDK sometimes there are simpler versions of what you want to do already included in the package, even if a bit hidden.
- The 4 scripts I present make it easier to build upon by delegating behaviors between scripts.
- Only 2 of them really need changing any changes being the GazeRecording the most important to alter has it uses Tobii technology.