DAFTAR PUSTAKA

Abdulrahaman, M. D., Faruk, N., Oloyede, A. A., Surajudeen-Bakinde, N. T., Olawoyin, L. A., Mejabi, O. V., Imam-Fulani, Y. O., Fahm, A. O., & Azeez, A. L. (2020). Multimedia tools in the teaching and learning processes: A systematic review. In *Heliyon* (Vol. 6, Issue 11). Elsevier Ltd. https://doi.org/10.1016/j.heliyon.2020.e05312

Alexis David Armijos Yunga, Nicole Adriana Juela Corte, Fabiana Valentina Pesantez Ocampo, & Manuel Estuardo Bravo Calderón. (2023). Comparative study of reliability in three software meshmixer, 3d slicer and nemocast of the intercanine and intermolar spaces of digital models. *World Journal of Advanced Research and Reviews*, *17*(1), 1040–1045. https://doi.org/10.30574/wjarr.2023.17.1.0111

Andrews, C., Southworth, M. K., Silva, J. N. A., & Silva, J. R. (2019). Extended Reality in Medical Practice. In *Current Treatment Options in Cardiovascular Medicine* (Vol. 21, Issue 4). Springer Healthcare. https://doi.org/10.1007/s11936-019-0722-7

Autodesk. (2018, April 15). *Mesh Mixer User Manual*. https://help.autodesk.com/view/MSHMXR/2019/ENU/

Blender. (2024, March 23). *Blender 4.1 Reference Manual*. https://docs.blender.org/manual/en/latest/index.html

Eslamipour, F., Borzabadi-Farahani, A., Le, B., & Shahmoradi, M. (2017). A retrospective analysis of dentofacial deformities and orthognathic surgeries. *Annals of Maxillofacial Surgery*, *7*(1), 73. https://doi.org/10.4103/ams.ams_104_16

Gregory Santos, & Mark W. Jones. (2023, May 29). *Prevention of Surgical Errors*. StatPearls. https://www.ncbi.nlm.nih.gov/books/NBK592394/

Hussain, F., Hussain, A., Shakeel, H., Uddin, N., & Ghouri, T. L. (2020). *Unity Game Development Engine: A Technical Survey*. http://sujo.usindh.edu.pk/index.php/USJICT/

Intel. (2018). *Demystifying the Virtual Reality Landscape*. https://www.intel.com/content/www/us/en/tech-tips-and-tricks/virtual-reality-vs-augmented-reality.html

Jacobs, L. M. (2023). *Study Analyzes Wrong-Site Surgery Data in Medical Malpractice Complaints*. https://www.facs.org/for-medical-professionals/news-publications/news-and-articles/bulletin/2023/june-2023-volume-108-issue-6/study-analyzes-wrong-site-surgery-data-in-medical-malpractice-complaints/

Jamiy, F. El, & Marsh, R. (2019). Distance estimation in virtual reality and augmented reality: A survey. *IEEE International Conference on Electro Information Technology*, *2019-May*, 063–068. https://doi.org/10.1109/EIT.2019.8834182

Johnson, S., Erdman, A. G., Jackson, B., Keefe, D. F., Tourek, B., & Molina, M. (2016). Immersive analytics for medicine: Hybrid 2D/3D sketch-based interfaces for annotating medical data and designing medical devices. *Companion Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces: Nature Meets Interactive Surfaces, ISS 2016*, 107–113. https://doi.org/10.1145/3009939.3009956

Karan R. Patil, Steven K. Ayer, Wei Wu, & Jeremi London. (2020). Mixed Reality Multimedia Learning to Facilitate Learning Outcomes from Project Based Learning. *Construction Research Congress 2020: Computer Applications*, 153–161. https://par.nsf.gov/servlets/purl/10275309

Kevin P. Pfeil, Sina Masnadi, Jacob Belga, Jose-Valentin T. Sera-Josef, & Joseph J. LaViola Jr. (2021). Distance Perception with a Video See-Through Head-Mounted Display. *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3411764.3445223

Kumar, R. P., Pelanis, E., Bugge, R., Brun, H., Palomar, R., Aghayan, D. L., Fretland, Å. A., Edwin, B., & Elle, O. J. (2020). Use of mixed reality for surgery planning: Assessment and development workflow. *Journal of Biomedical Informatics: X*, *8*. https://doi.org/10.1016/j.yjbinx.2020.100077

Lee, R., Goonewardene, M. S., Mian, A., Allan, B., Brock, D., & Trevenen, M. (2018). Accuracy of orthognathic surgery using 3D computer-assisted surgical simulation. In *Australasian Orthodontic Journal* (Vol. 34, Issue 1).

Lu, L., Wang, H., Liu, P., Liu, R., Zhang, J., Xie, Y., Liu, S., Huo, T., Xie, M., Wu, X., & Ye, Z. (2022). Applications of Mixed Reality Technology in Orthopedics Surgery: A Pilot Study. *Frontiers in Bioengineering and Biotechnology*, *10*. https://doi.org/10.3389/fbioe.2022.740507

Meta. (2022). *This is Meta Quest Pro*. https://www.meta.com/quest/quest-pro/#overview

Microsoft. (2022). *Mixed Reality Documentation*. https://learn.microsoft.com/en-us/windows/mixed-reality/

Milgram, P., & Kishino, F. (1994). A Taxonomy of Mixed Reality Visual Displays Unconscious Computing View project Augmented Reality through Graphic Overlays on Stereoscopic video View project A TAXONOMY OF MIXED REALITY VISUAL DISPLAYS. In *IEICE Transactions on Information Systems* (Issue 12). http://vered.rose.utoronto.ca/people/paul_dir/IEICE94/ieice.html

Naran, S., Steinbacher, D. M., & Taylor, J. A. (2018). Current concepts in orthognathic surgery. *Plastic and Reconstructive Surgery*, *141*(6), 925e–936e. https://doi.org/10.1097/PRS.0000000000004438

Oculus. (2024, April). *Oculus Documentation*. https://developer.oculus.com/documentation

Parveau, M., & Adda, M. (2018). 3iVClass: A new classification method for virtual, augmented and mixed realities. *Procedia Computer Science*, *141*, 263–270. https://doi.org/10.1016/j.procs.2018.10.180

Patel, P. K., & Novia, M. V. (2007). The Surgical Tools: The LeFort I, Bilateral Sagittal Split Osteotomy of the Mandible, and the Osseous Genioplasty. In *Clinics in Plastic Surgery* (Vol. 34, Issue 3, pp. 447–475). https://doi.org/10.1016/j.cps.2007.05.012

Paula R. Patel, & Orlando De Jesus. (2023, January 2). *CT Scan*. StatPearls. https://www.ncbi.nlm.nih.gov/books/NBK567796/

Rahman, R., Wood, M. E., Qian, L., Price, C. L., Johnson, A. A., & Osgood, G. M. (2020). Head-Mounted Display Use in Surgery: A Systematic Review. In *Surgical Innovation* (Vol. 27, Issue 1, pp. 88–100). SAGE Publications Inc. https://doi.org/10.1177/1553350619871787

Roedavan, R., Pudjoatmodjo, B., & Putri Sujana, A. (2022). *MULTIMEDIA DEVELOPMENT LIFE CYCLE (MDLC)*. https://doi.org/10.13140/RG.2.2.16273.92006

Ruslin, M., Forouzanfar, T., Astuti, I. A., Soemantri, E. S., & Tuinzing, D. B. (2015). The epidemiology, treatment, and complication of dentofacial deformities in an Indonesian population: A 21-year analysis. *Journal of Oral and Maxillofacial Surgery, Medicine, and Pathology*, *27*(5), 601–607. https://doi.org/10.1016/j.ajoms.2014.09.006

Sánchez-Margallo, J. A., Plaza de Miguel, C., Fernández Anzules, R. A., & Sánchez-Margallo, F. M. (2021). Application of Mixed Reality in Medical Training and Surgical Planning Focused on Minimally Invasive Surgery. *Frontiers in Virtual Reality*, *2*. https://doi.org/10.3389/frvir.2021.692641

Saravia-Rojas, M., Gutiérrez-Trevejo, J., Fukuhara-Nakama, M., & Velásquez- Huaman, Z. (2021). Autodesk Meshmixer usado en la

enseñanza de la odontología: Es posible? *Revista Estomatológica Herediana*, *31*(4), 323–329. https://doi.org/10.20453/reh.v31i4.4102

Shetty, S. K., Neeraja, & Yethadka, M. (2017). CBCT in Orthognathic Surgery. *Scholars Journal of Dental Sciences (SJDS)* , 547–555. https://doi.org/10.21276/sjds.2017.4.12.4

Skarbez, R., Smith, M., & Whitton, M. C. (2021). Revisiting Milgram and Kishino's Reality-Virtuality Continuum. *Frontiers in Virtual Reality*, *2*. https://doi.org/10.3389/frvir.2021.647997

Speicher, M., Hall, B. D., & Nebeling, M. (2019, May 2). What is mixed reality? *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3290605.3300767

Unity. (2020, June 5). *Preparing Assets for Unity*. https://docs.unity3d.com/2019.3/Documentation/Manual/BestPracticeMakingBelievableVisuals1.html.

Unity. (2023, November). *Using Blender and Maya with Unity*. https://unity.com/how-to/beginner/using-blender-and-maya-unity

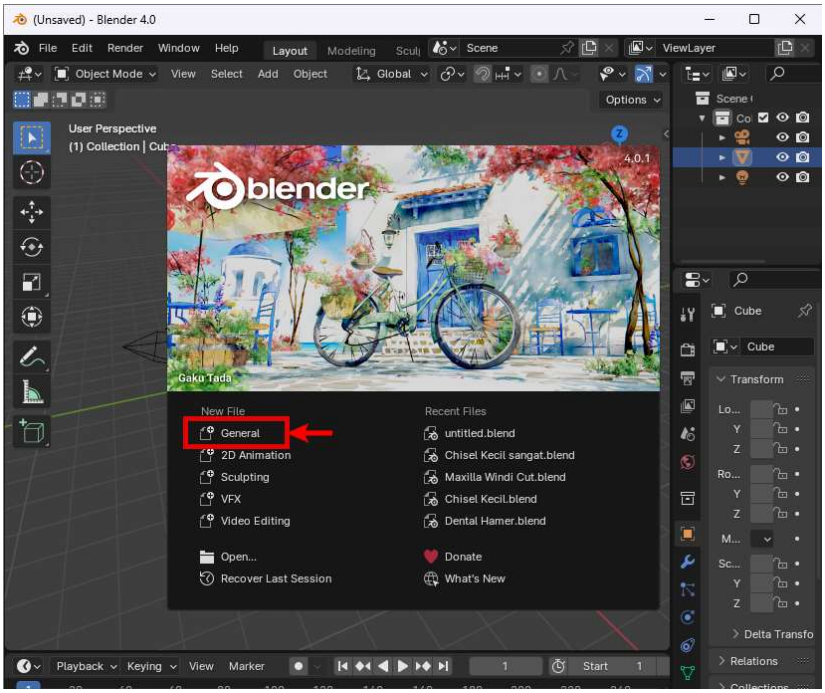
Yudho Yudhanto, & Anggi Sulistiawan. (2022). *Panduan aplikasi virtual reality (VR)* (1st ed.). PT. Elex Media Komputindo.

LAMPIRAN

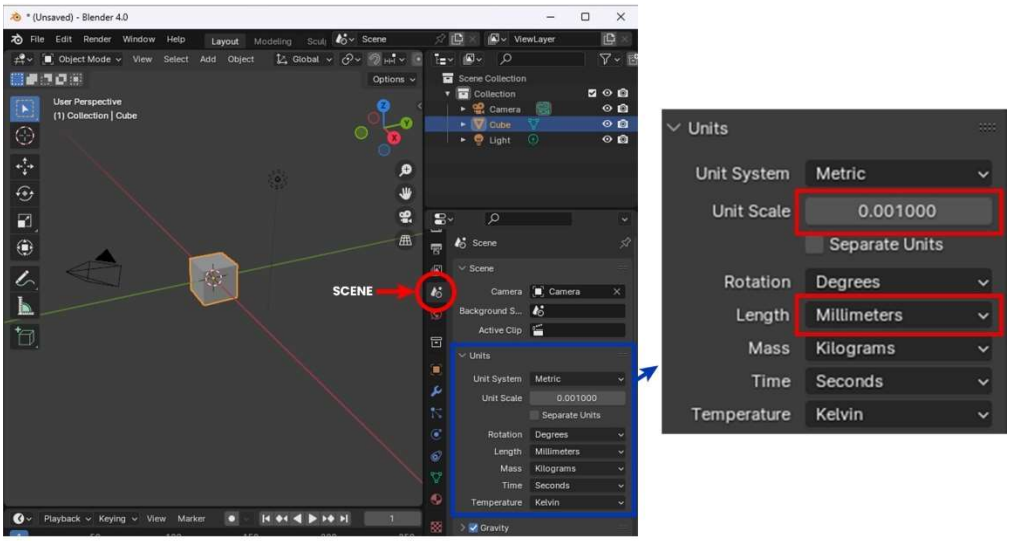
Lampiran 1 Menyiapkan Objek 3D

a. Menyiapkan *template file* blender

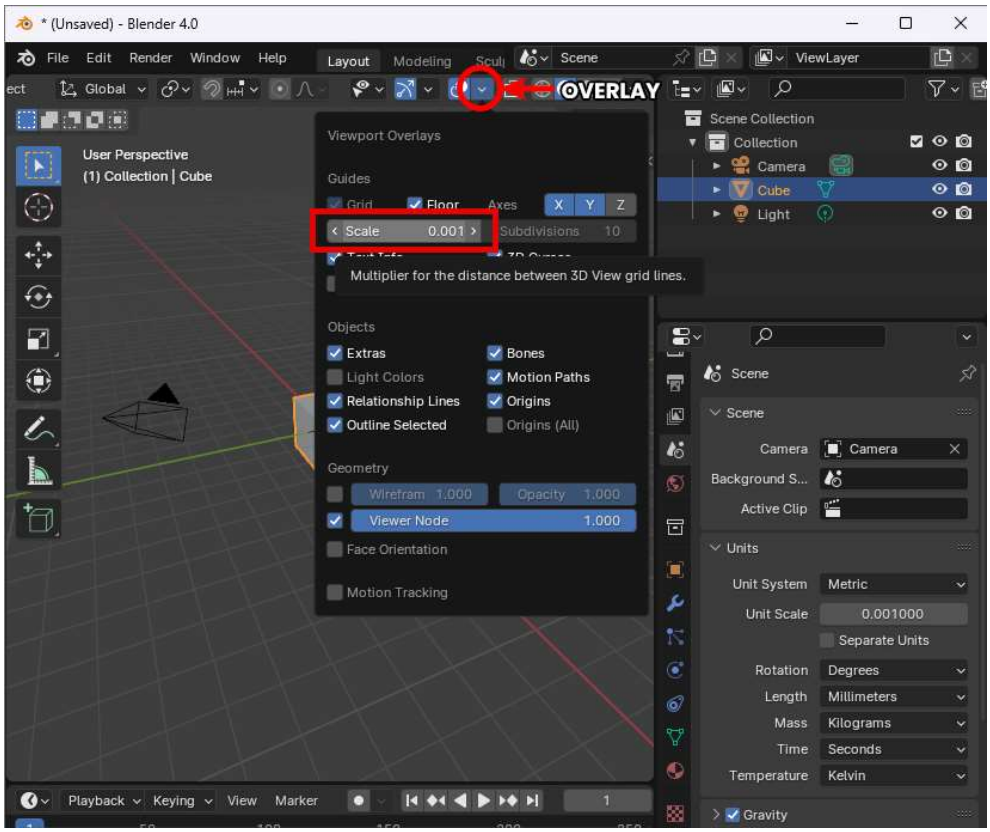
1. Langkah pertama yang dilakukan yaitu membuka aplikasi Blender 4.0, kemudian memilih opsi *General* yang tertera pada tampilan awal aplikasi.



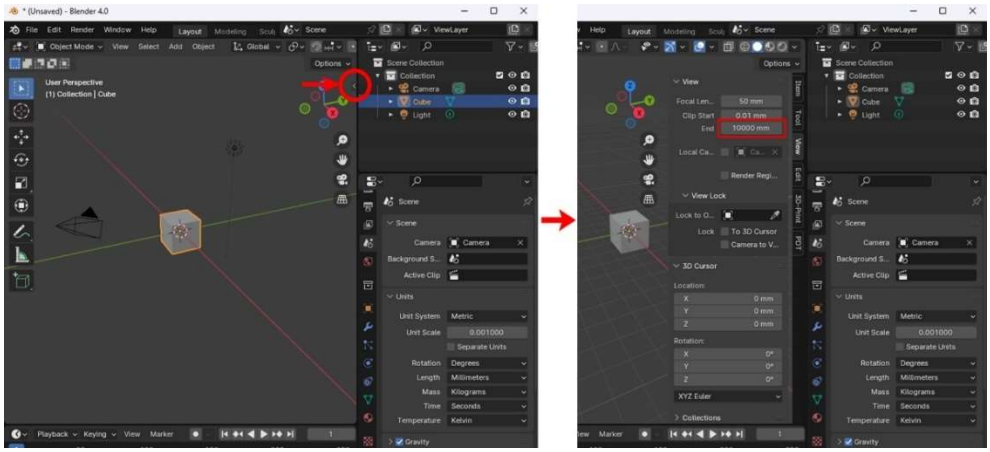
2. Buka kolom *Scene* pada bagian kanan, kemudian pada bagian *Units* ubah *Unit Scale* dari 1.0 menjadi 0.001 dan *Leght* dari *Meters* ke *Milimeters*.



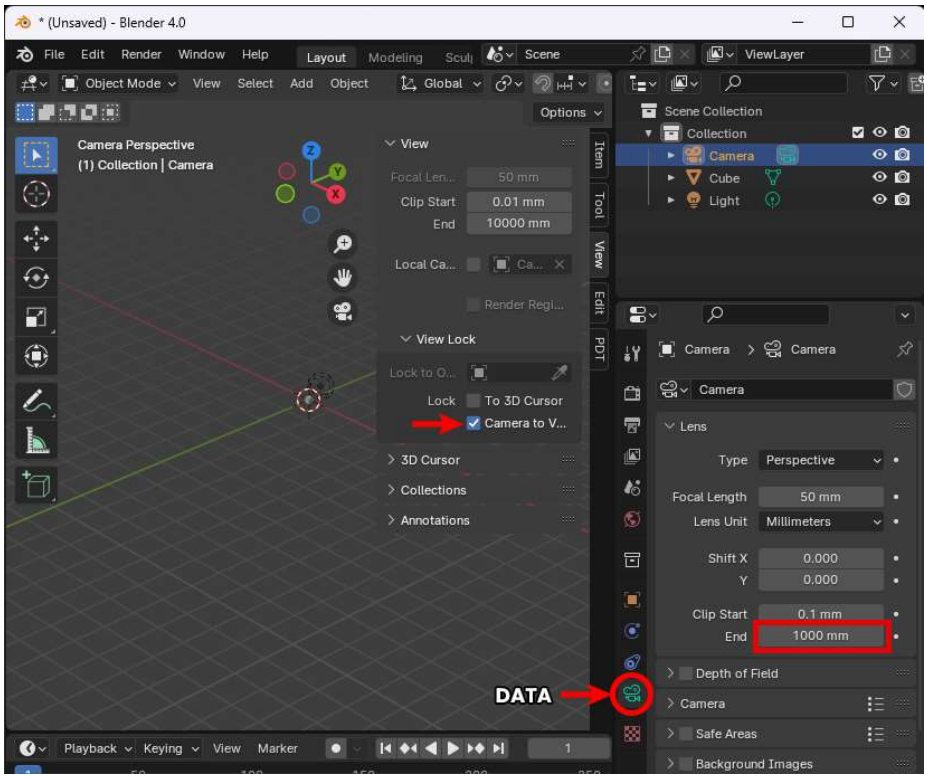
3. Buka kolom *Overlay* pada bagian atas, kemudian ganti *scale* dari 1.0 menjadi 0.001.



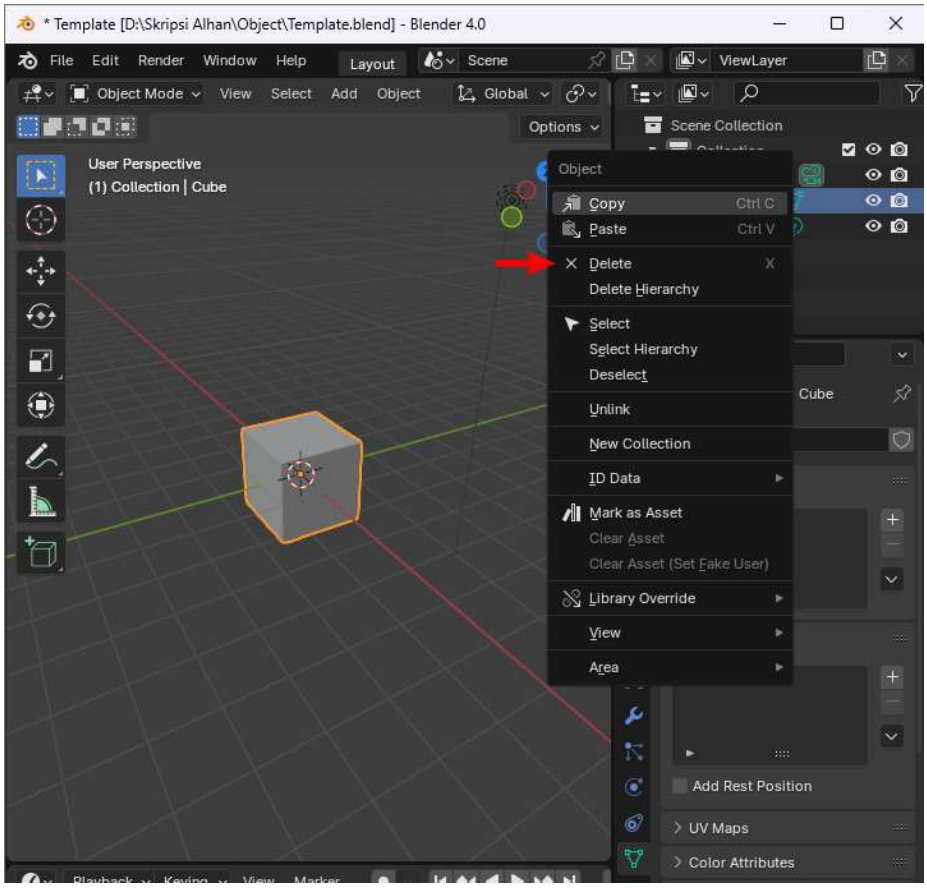
4. Tekan tanda panah yang berada pada bagian kiri *Scene Collection*, kemudian pada bagian *View* ubah kolom *End* dari 1000 mm menjadi 10000 mm.



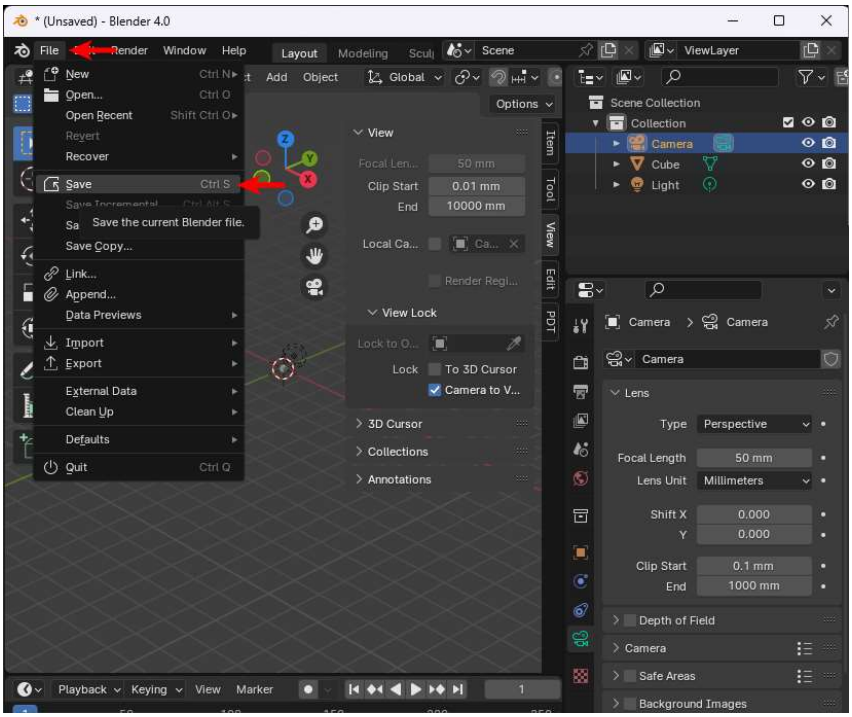
5. Tekan "0" (nol) pada keyboard, kemudian centang bagian *Camera to View*. Setelah itu buka bagian *data* pada bagian kanan dan ubah nilai kolom *End* dari 100 mm menjadi 1000 mm.



6. Hapus kubus pada *scene* dengan klik kanan *cube* pada bagian *scene selection* dan pilih *delete*.

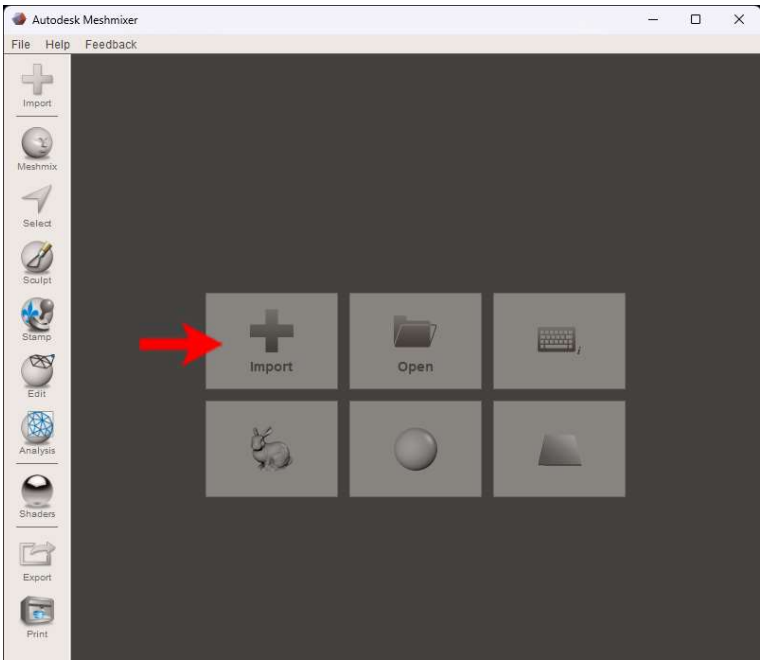


7. Simpan *file template* ini dengan cara pilih *File* > *Save* atau menekan tombol *Ctrl* + *S* pada keyboard.

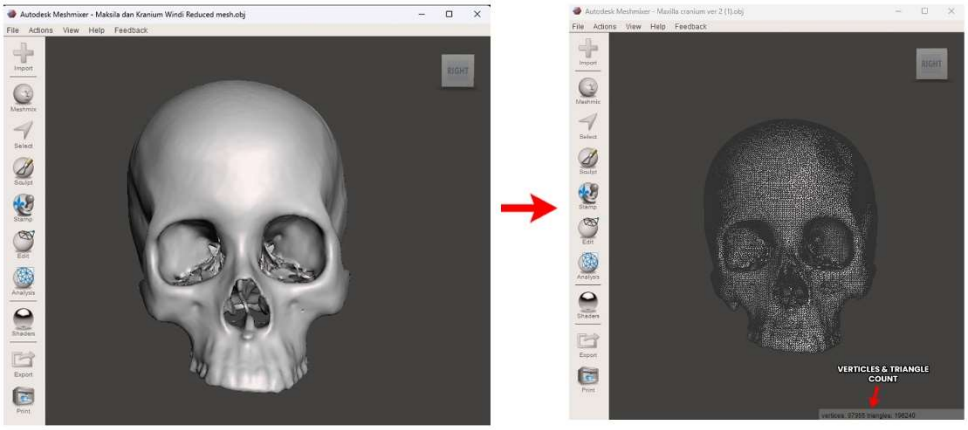


b. Menyiapkan *file* 3D rahang

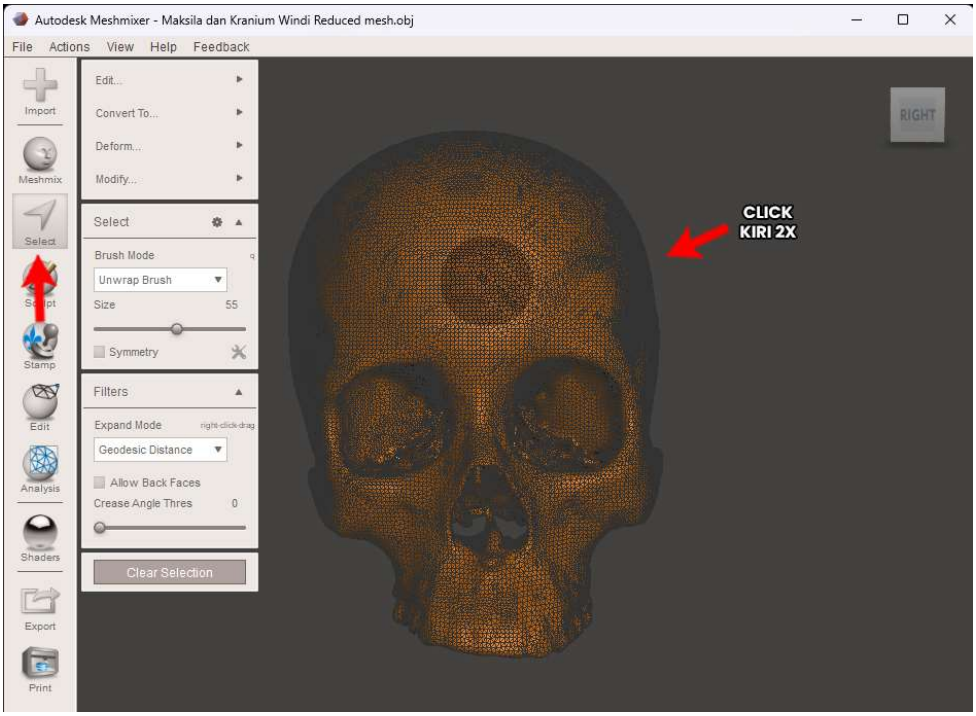
1. Langkah pertama yang dilakukan yaitu membuka aplikasi Meshmixer (versi 3.5.474), kemudian pilih *Import* dan masukkan *file* format 3D rahang hasil CT scan yang berformat .STL.



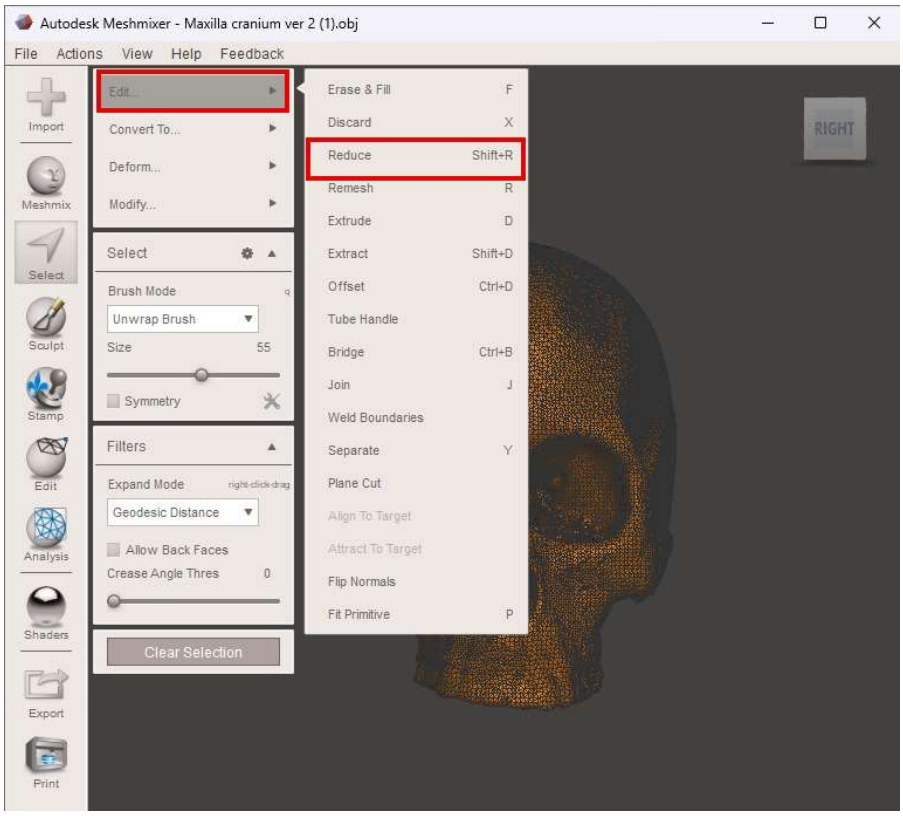
2. Tekan tombol “W” di keyboard untuk mengaktifkan mode *wayframe* yang berguna untuk memudahkan proses seleksi ke depannya. Jumlah *vertices* dan *triangle* pada Objek 3D dapat dilihat pada bagian kanan bawah layar.



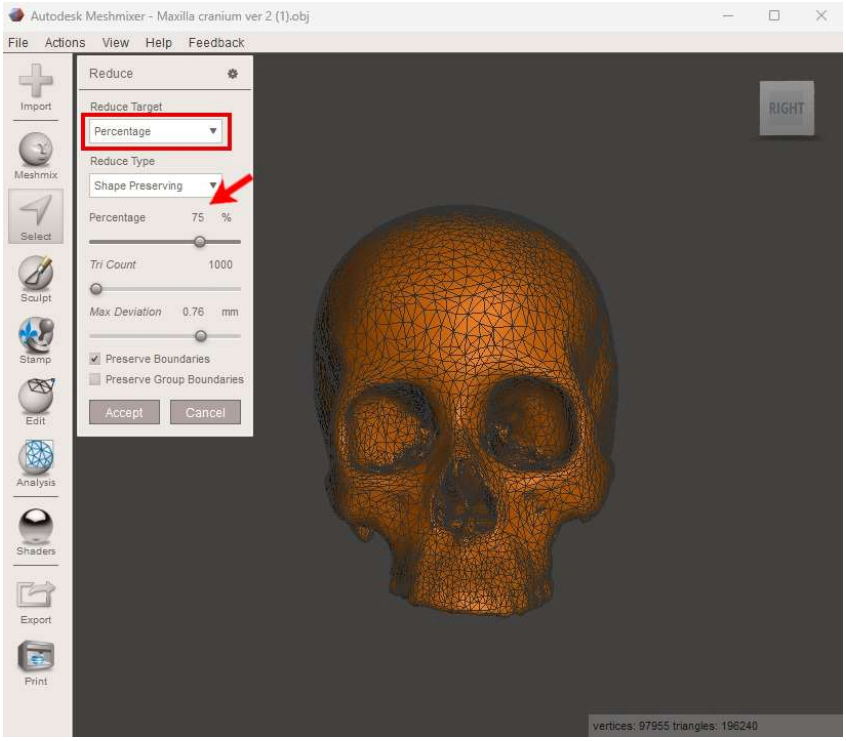
3. Pilih “*Select*” pada menu yang berada di bagian kiri layar, kemudian arahkan kursor ke objek 3D dan klik kiri 2x pada mouse untuk menandai semua area objek 3D.



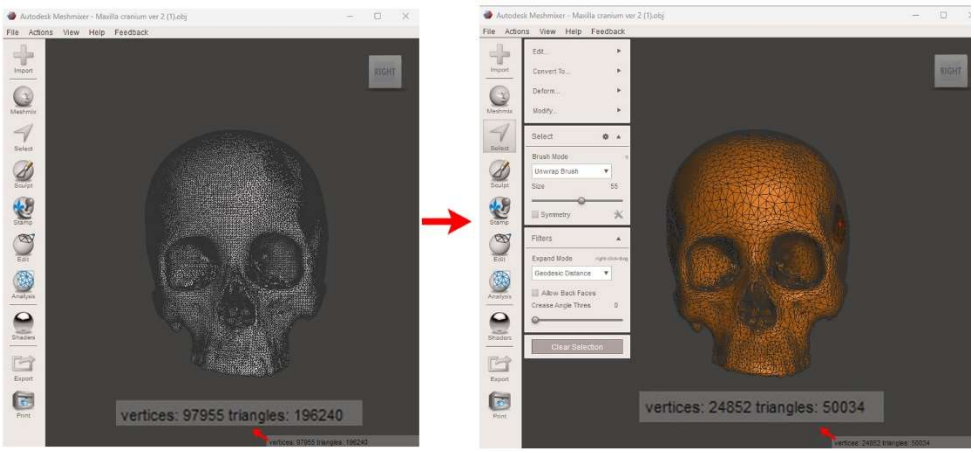
4. Arahkan kursor ke menu “*Edit*” dan kemudian. pilih “*Reduce*” atau klik “*Shift* + *R*” pada keyboard.



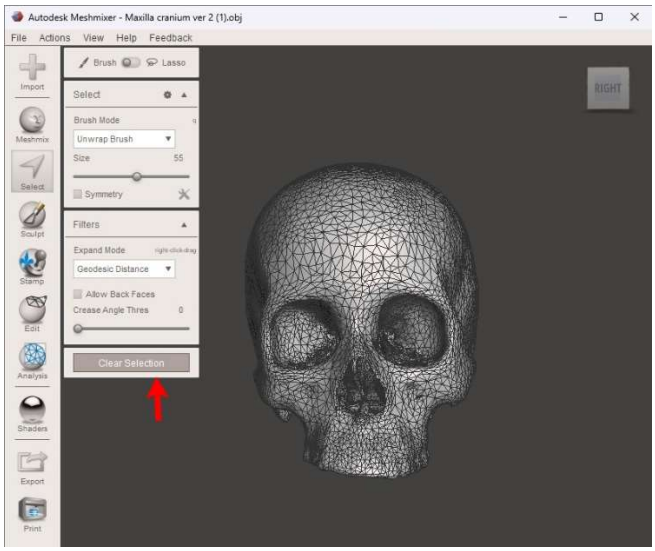
5. Atur reduce target menjadi "*Percentage*", kemudian atur "*Percentage*" menjadi 75% dan pilih "*Accept*".



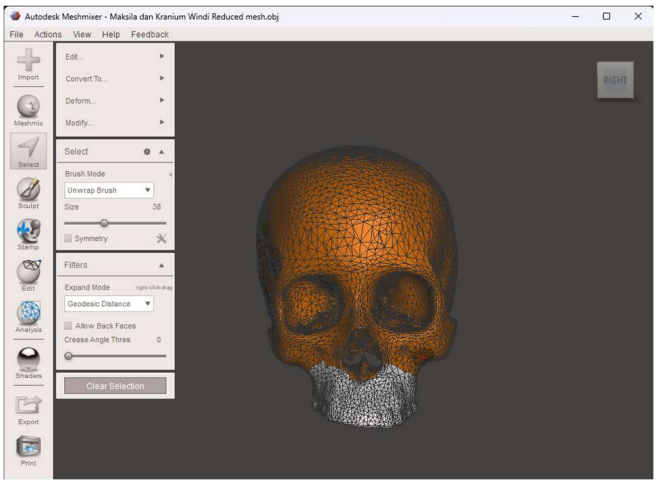
6. Sekarang dapat dilihat perubahan yang terjadi pada jumlah *vertices* dan *triangles*.



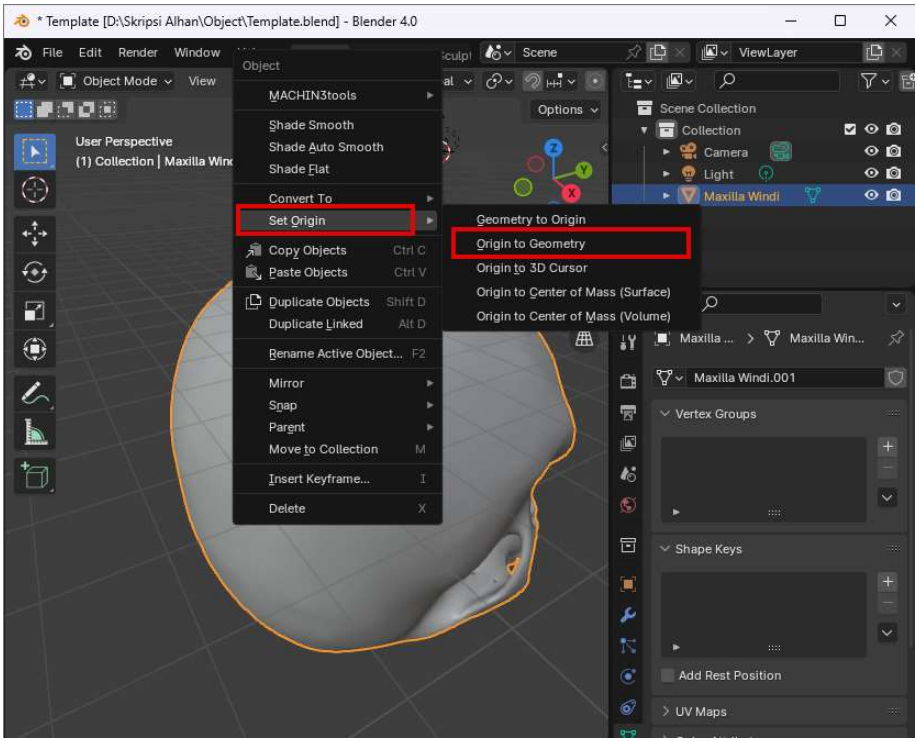
7. Tekan "*clear selection*" untuk menghilangkan *selection* pada objek 3D.



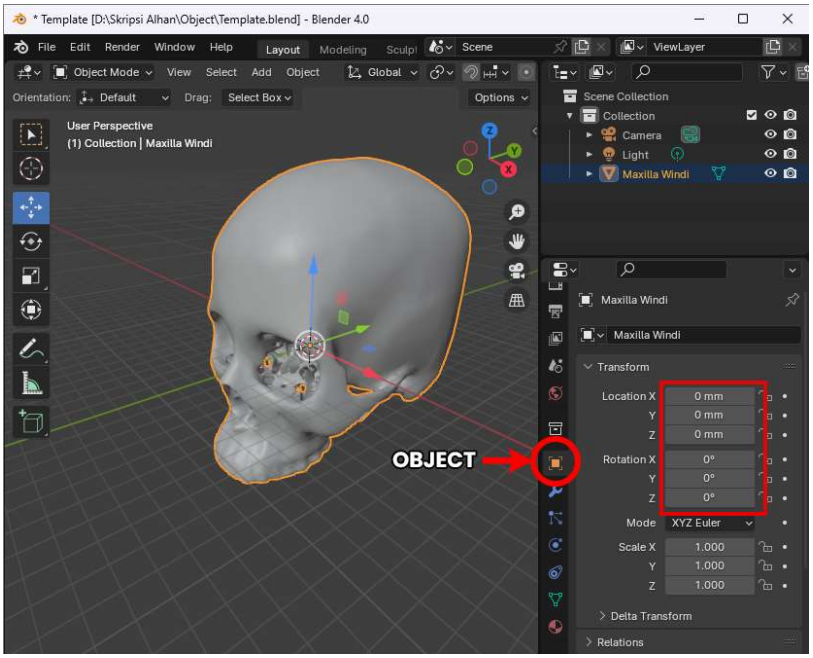
8. Kali ini lakukan seleksi manual dengan menu *select* dan lakukan seleksi pada bagian objek 3D yang detailnya tidak diperlukan. Pada contoh gambar di bawah, detail yang diperlukan berada pada bagian bawah tengkorak sehingga kita menandai bagian atasnya.



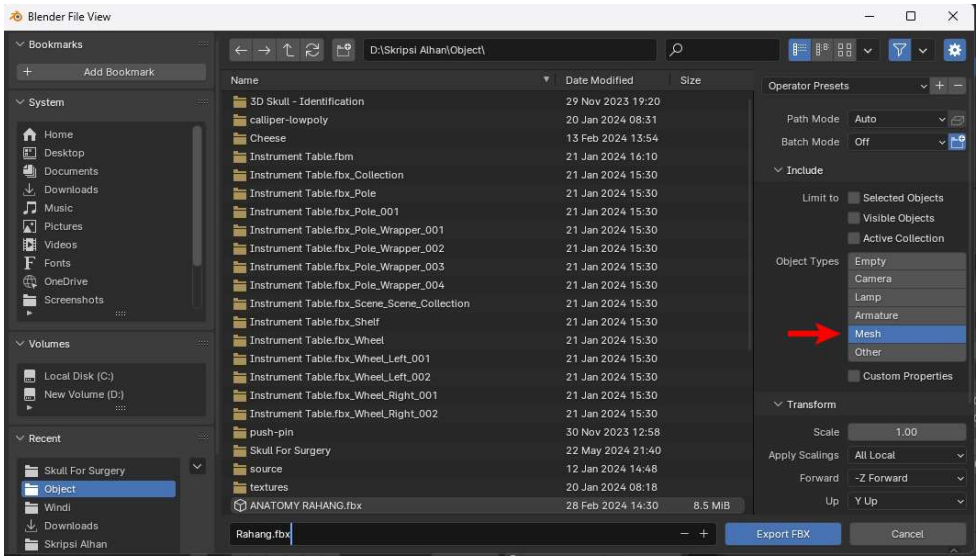
9. Lakukan hal sama seperti langkah kelima namun kali ini ubah *percentage*-nya menjadi 50% saja.
10. *Export file* dengan format .OBJ dengan cara klik *File* > *Export* atau tekan "*Ctrl* + *E*" pada keyboard.
11. Buka *file template* Blender yang sebelumnya telah dibuat dan *import* masuk file .OBJ dari Meshmixer yang dibuat.
12. Klik kanan pada objek 3D dan kemudian pilih Set *Origin* > *Origin to Geometry* untuk memperbaiki posisi titik tengah objek yang melenceng.



13. Klik menu "Objek" pada bagian kanan layar dan ubah parameter *Location* XYZ dan *Rotation* XYZ menjadi 0. Sekarang objek 3D sudah berada diposisi yang seharunya.



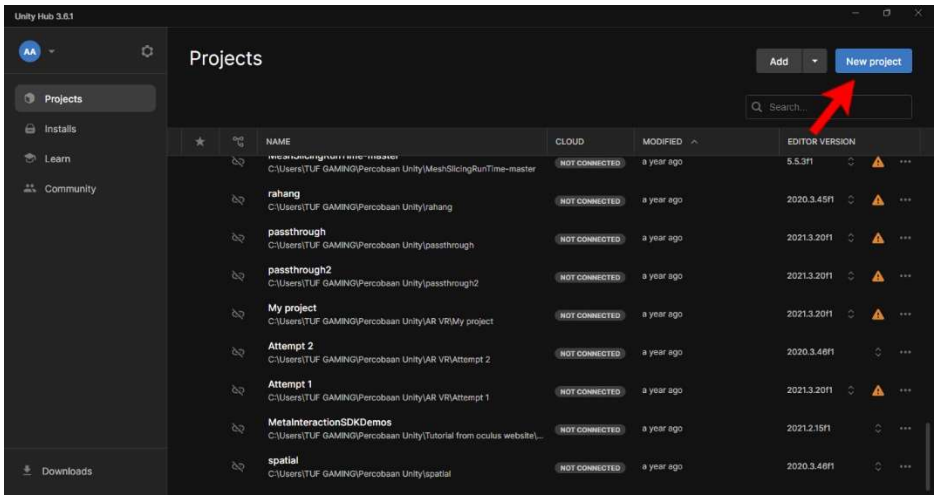
14. *Export file* menjadi format .FBX dengan klik *File* > *Export* > *FBX (.fbx)* dan kemudian tandai saja bagian *mesh* agar yang ter-*export* cuman objeknya saja.



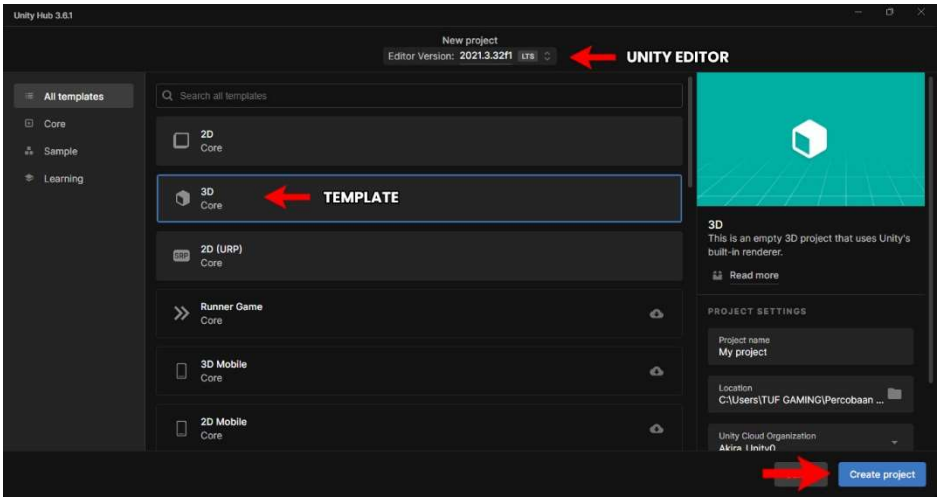
Lampiran 2 Menyiapkan *File* Unity

a. Menyiapkan Unity

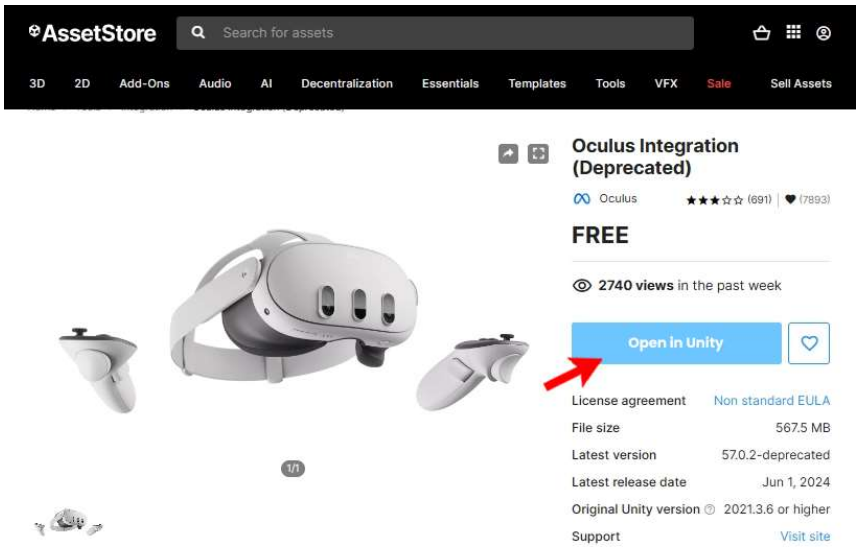
1. Langkah pertama yang dilakukan yaitu membuka aplikasi Unity Hub, kemudian pilih *New Project.*



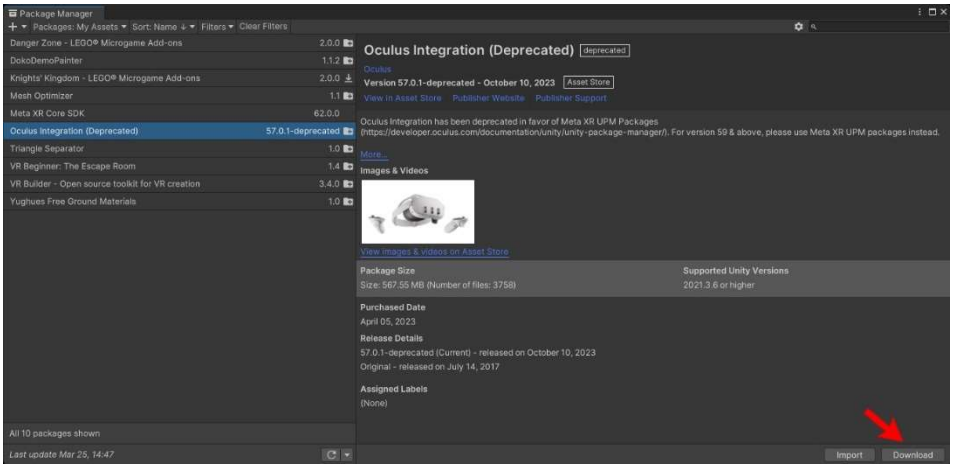
2. Pilih Unity Editor versi 2021.3.32f1 LTS dan *3D Core* sebagai *template* awal pembuatan aplikasi, kemudian tekan *Creat Project*.



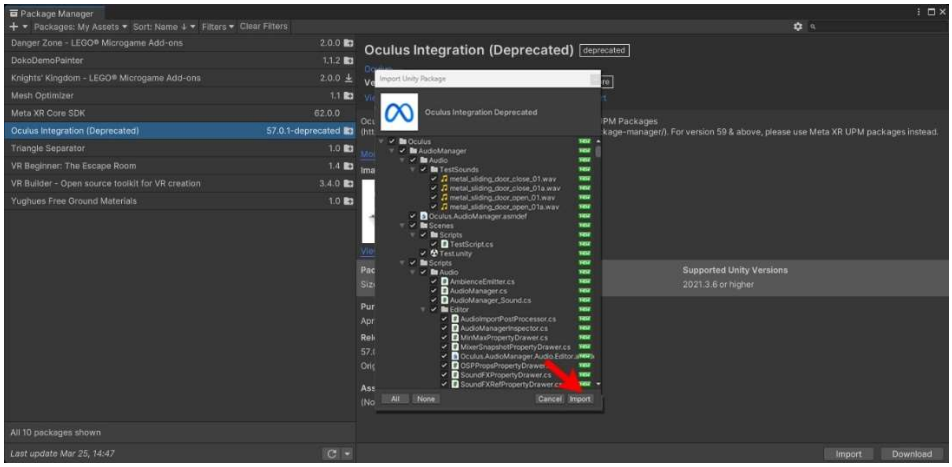
3. Buka *https://assetstore.unity.com/* di browser, masuk menggunakan akun Unity yang sama dengan di aplikasi, lalu cari “Oculus Integration”, kemudian klik *Open in Unity*. Unity akan membuka jendela *Package Manager* di editor Unity.



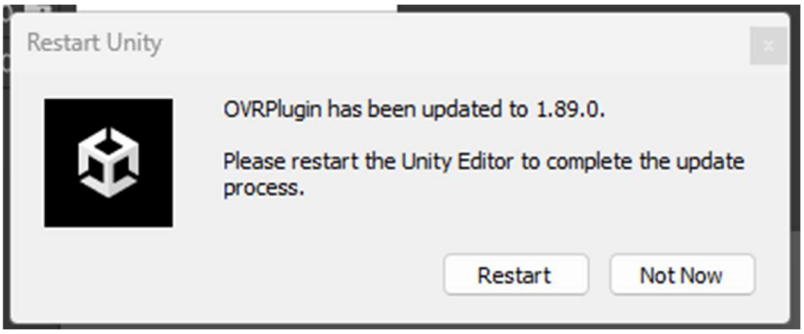
4. Pada jendela *Package Manager*, perluas Oculus Integration, pilih versi terbaru, lalu di sisi kanan bawah jendela, klik *Download*.



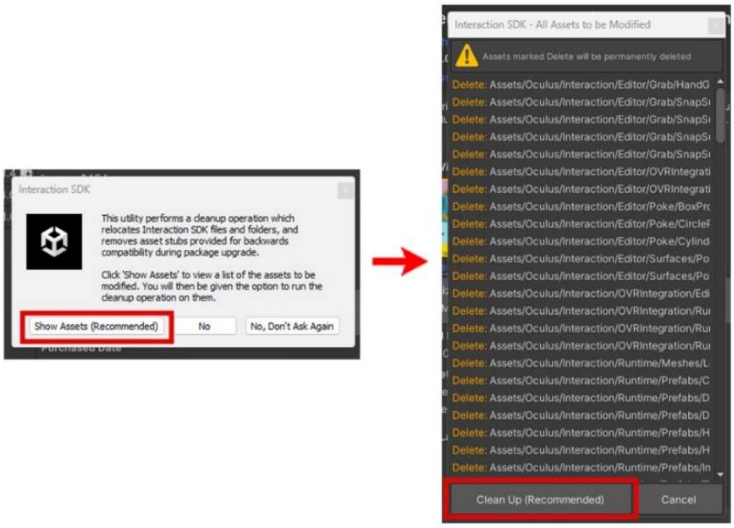
5. Setelah pengunduhan selesai, klik *Import* untuk mengimpor Oculus Integration terbaru ke dalam *Project* yang dibuat sebelumnya.



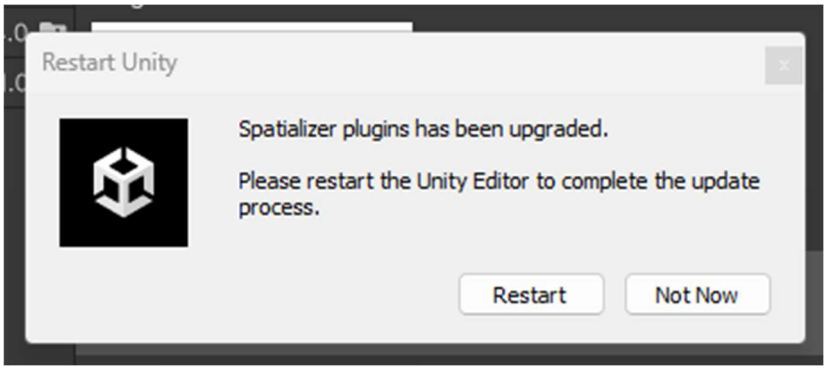
6. Pada jendela *Import Unity Package*, biarkan semua *file* dan folder dipilih, dan klik *Import*.
7. Ketika diminta untuk memperbarui *plugin Oculus Utilities*, klik *Restart*. Langkah ini memastikan untuk menggunakan *plugin* yang disertakan dengan paket yang di-*instal*. Jika memilih untuk tidak memperbarui *plugin* pada saat ini, diperlukan pembaruan secara manual nanti.



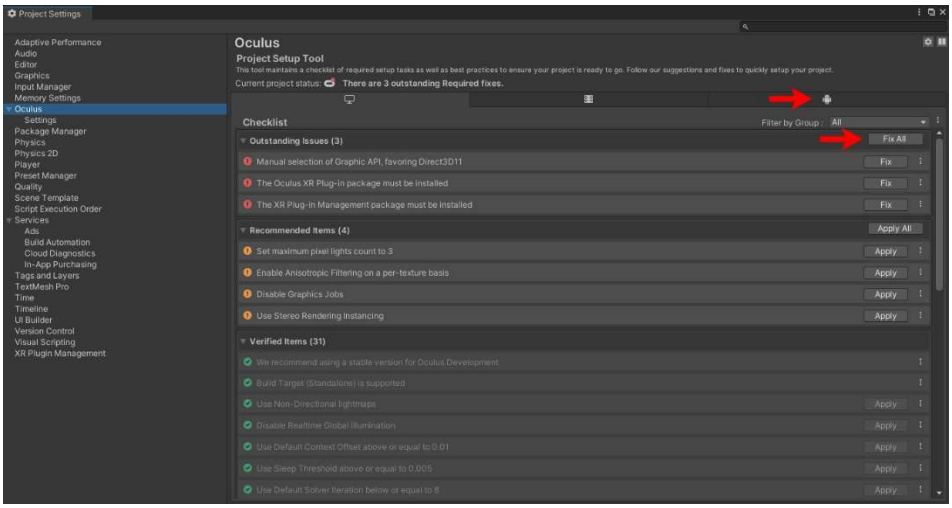
8. Saat diminta untuk membersihkan aset lama, klik *Show Assets*, lalu klik *Clean Up*.



9. Saat diminta untuk memperbarui *plugin Spatializer*, klik *Restart*.

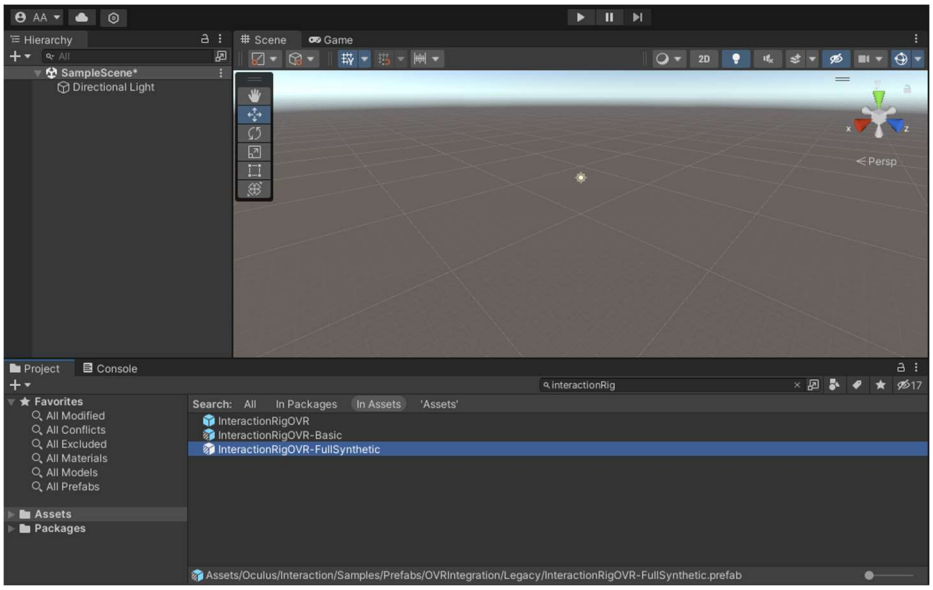


10. Setelah *restart* selesai, pergi ke *File* > *Build Setting* > *Player Setting* > *Oculus*, kemudian tekan *Fix All*. Kemudian buka tab dengan logo Android dan tekan *Fix All*.



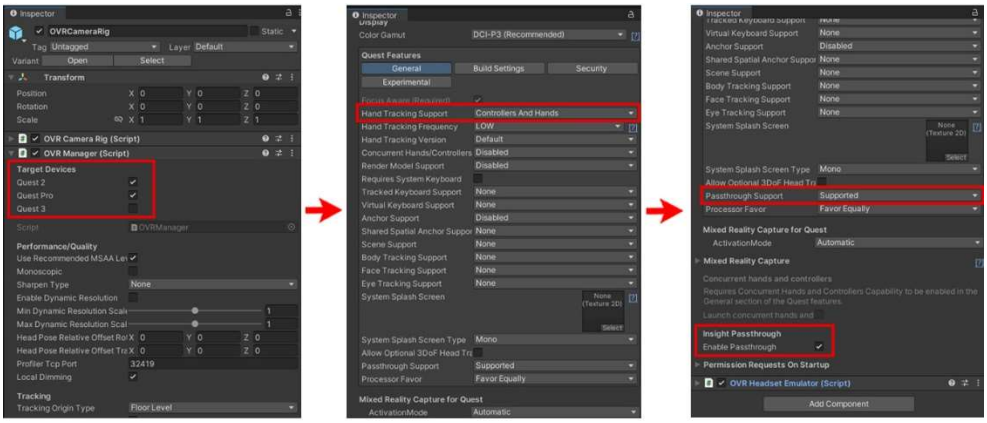
b. Mengaktifkan *Passthrough*

1. Pilih *Main Camera* di tab *Hierarchy* dan hapus.
2. Pada tab *Project*, cari *InteractionRigOVR-FullSynthetic*, lalu seret ke dalam tab *Hierarchy*.

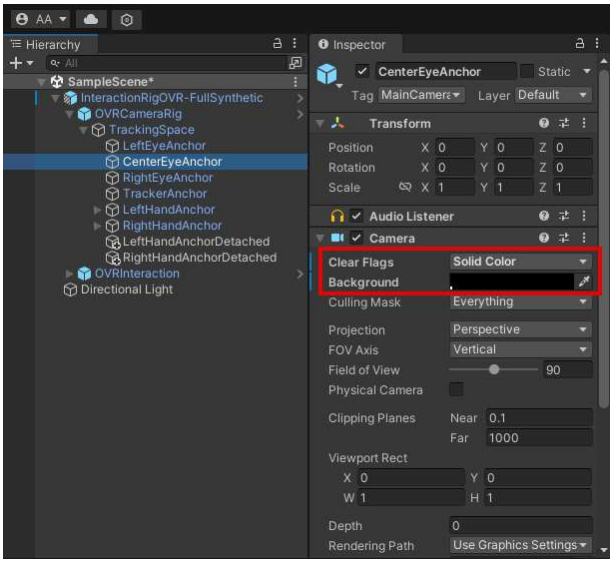


3. Pada tab *Hierarchy*, perluas *InteractionRigOVR-FullSynthetic* dengan menekan tanda panah di kirinya, kemudian pilih *OVRCameraRig*,
4. Pada tab *Inspector*, di bawah *OVR Camera Rig* perluas *OVR Manager*, lakukan hal berikut:

- Pada *Target Devices*, centang Quest 2 dan Quest Pro.
- Di bawah *Quest Features* > *General tab*, ubah *Hand Tracking Support* dari *Controllers Only* ke *Controllers And Hands*.
- Masih *General tab*, ubah *Passthrought Support* dari *None* ke *Supported*.
- Pada bagian *Mixed Reality Capture*, centang *Enable Passthrought*

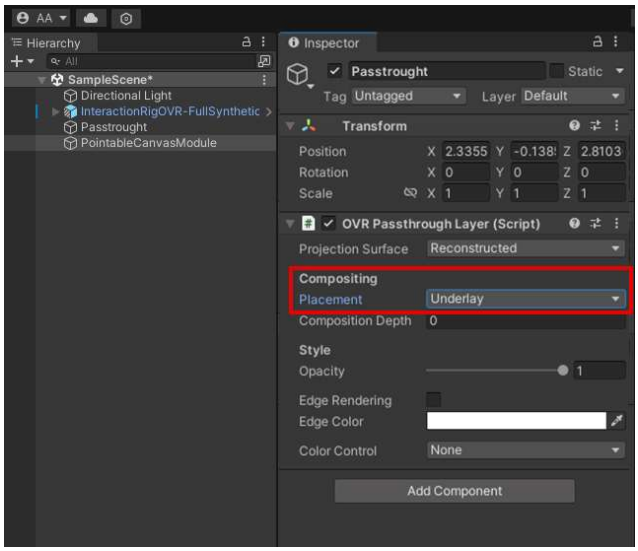


5. Pada tab *Hierarchy*, perluas *OVRCameraRig*, setelah itu perluas lagi *Traking Space* dan pilih *Center Eye Anchor*.
6. Pada tab *Inspector*, di bawah *Audio Listener* perluas *Camera*, lakukan hal berikut:
   - Ubah *Clear Flags* dari *Skybox* menjadi *Solid Color*.
   - Ubah warna *Background* menjadi warna hitam.



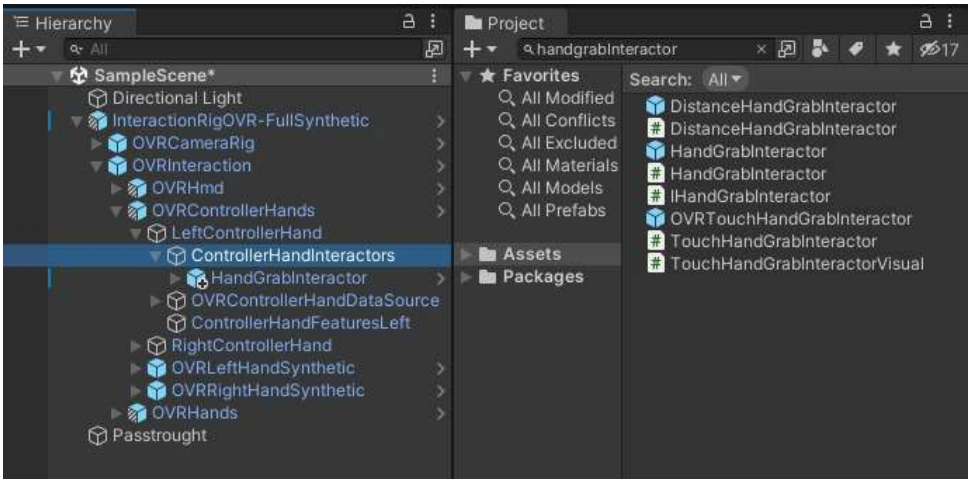
7. Buat *GameObject* baru dengan klik kanan di tab *Inspector* dan memilih *Create Empty* atau menekan logo "+" di pojok kiri atas tab *Inspector* dan namakan *Passthrought*.

8. Pada tab *Inspector* dari *Passthrought Layer*, klik *Add Component*. Lalu pada *Scripts*, pilih *OVR Passthrough Layer*. Kemudian pada bagian *Compositing* ubah *Placement* dari *Overlay* menjadi *Underlay*.

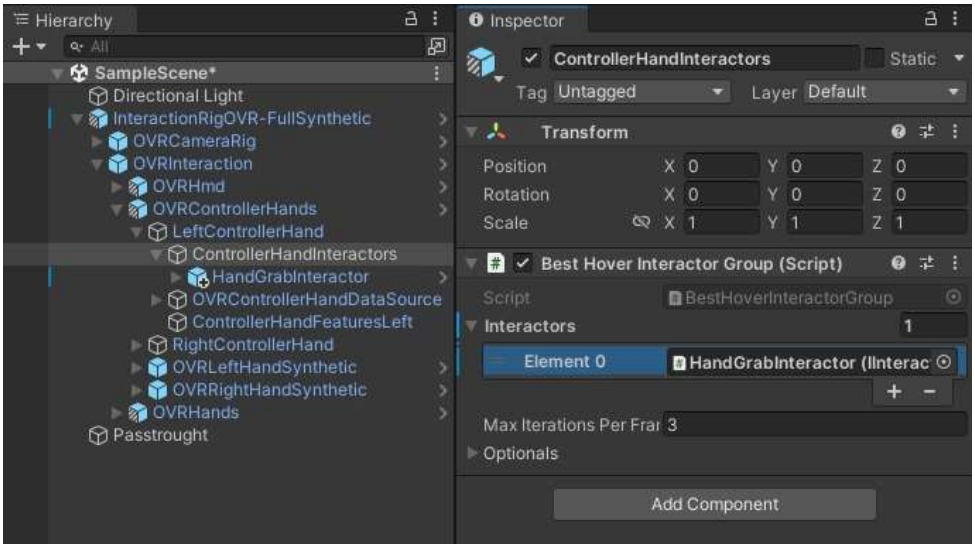


c. Memegang Objek

1. Pada bagian pencarian di tab *Project*, cari *HandGrabInteractor*. Pastikan filter pencarian diatur ke *All* atau *In Packages*, karena pengaturan *default* hanya mencari dalam *In Assets*.
2. Seret *prefab HandGrabInteractor* dari hasil pencarian ke dalam tab Hierarchy ke *InteractionRigOVR-FullSynthetic* > *OVRInteraction* > *OVRControllerHands* > *LeftControllerHands* > *ControllerHandInteractors*.

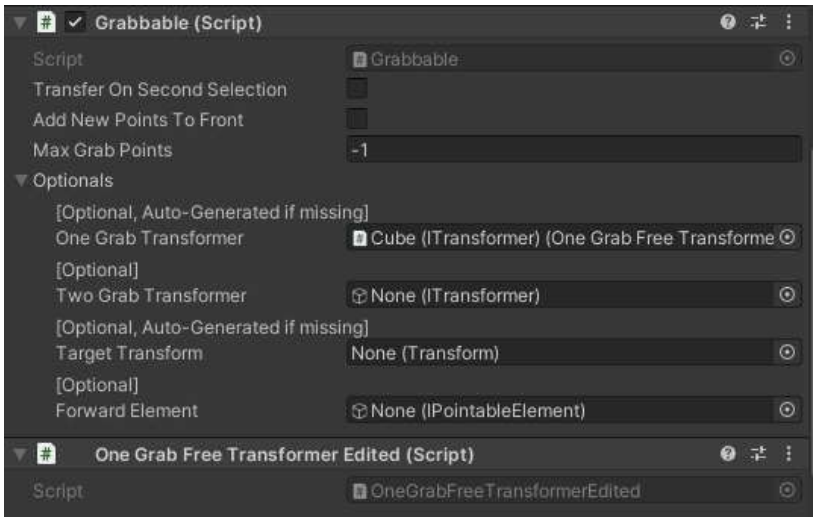


3. Tekan *LeftControllerHands* dan pada tab *Inspector*, terdapat komponen *Best Hover Interactor Group*, klik tanda + untuk menambahkan elemen ke daftar *Interactors*.
4. Seret *HandGrabInteractor* pada tab *Hierarchy* yang berada di bawah *LeftControllerHands* ke dalam kolom *Interactors* pada *Best Hover Interactor Group*.

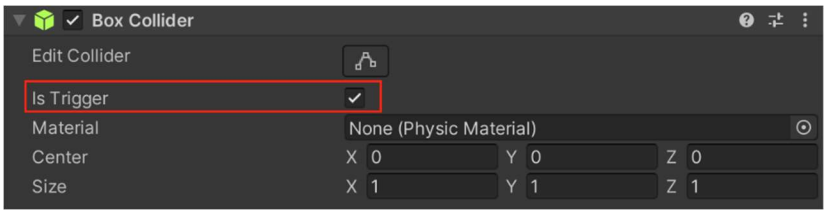


5. Lakukan langkah 2-4 untuk membuat interaksi *grab* pada *RightControllerHands*, juga pada *LeftHand*, dan *RightHand* yang berada di *OVRHands*.
6. Tambahkan *GameObject* Kubus (*Cube*) ke *scene* dengan mengklik kanan pada *Hierarchy* dan memilih Objek 3D > Kubus.
7. Pada *Hierarchy*, pilih *Cube* dan dalam tab *Inspector*, di komponen *Transform*, atur X, Y, dan Z dari properti *Scale* ke 0.1.
8. Dalam *scene*, posisikan *Cube* sehingga berada di depan kamera.
9. Pada *Hierarchy*, pilih *Cube*, dalam tab *Inspector*, tambahkan komponen *RigidBody* dan *Grabbable*. Komponen *Grabbable* adalah komponen yang menyebabkan objek yang dipilih bergerak.
10. Pada komponen *Grabbable*, pilih kotak *Transfer on Second Selection*. Ini memungkinkan untuk memindahkan objek di antara kedua tangan saat dipegang.
11. Kemudian tambahkan lagi komponen *OneGrabFreeTransformerEdited* pada *Cube* dan seret komponen ini ke dalam kolom *One Grab Transformer*

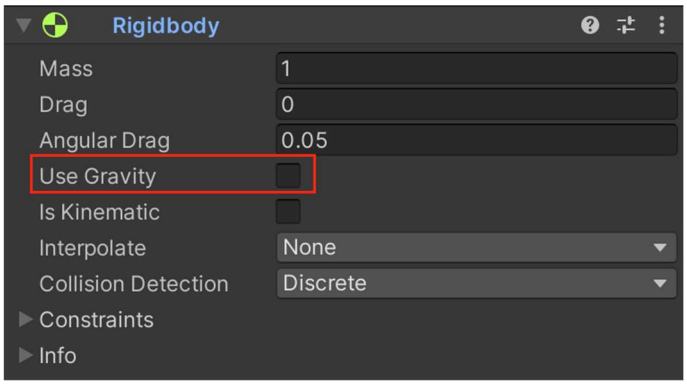
yang berada pada komponen *Grabbable*. Komponen ini yang digunakan sebagai parameter untuk mengetahui kondisi benda apa yang sedang dipegang.



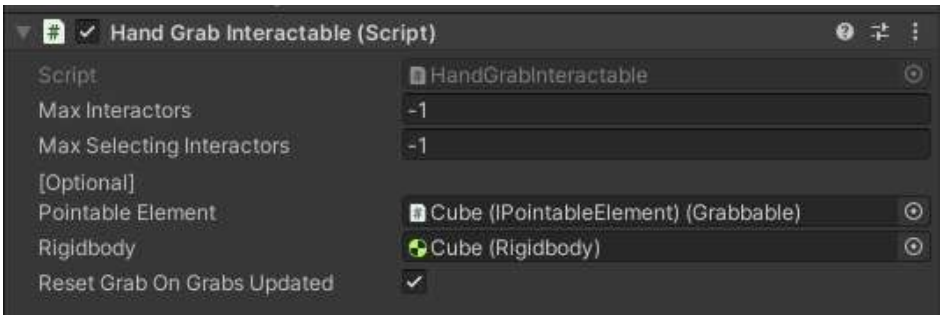
12. Pada komponen *Box Collider*, pilih kotak centang *Is Trigger*. Hal ini akan menghentikan kubus agar tidak melayang saat dilepaskan.



13. Pada komponen *Rigidbody*, hapus centang pada kotak centang *Use Gravity*.

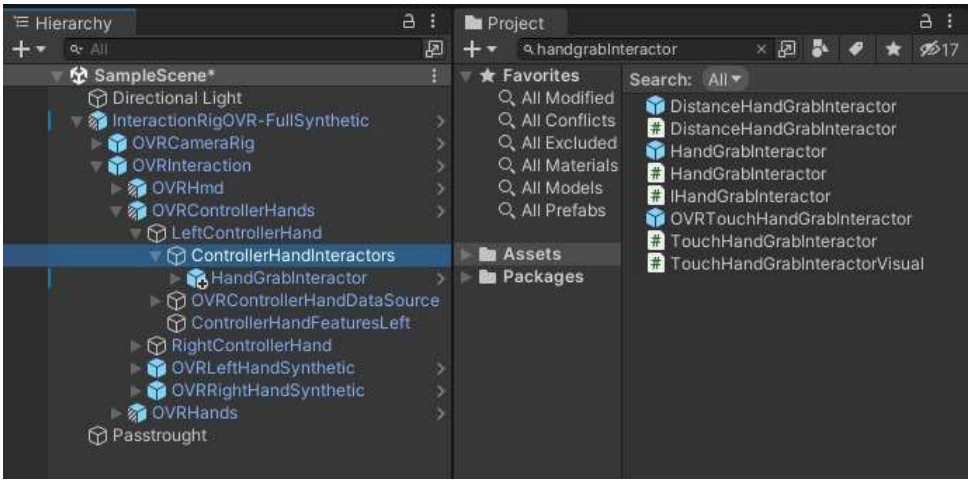


14. Tambahkan lagi komponen *Hand Grab Interactable* pada *Cube*. Apabila komponen ini ditambahkan sebelum menambahkan komponen *Grabable* dan *Rigidbody*, maka kolom *Pointable Element* dan *Rigidbody* pada komponen *Hand Grab Interactable* harus ditambahkan secara manual.

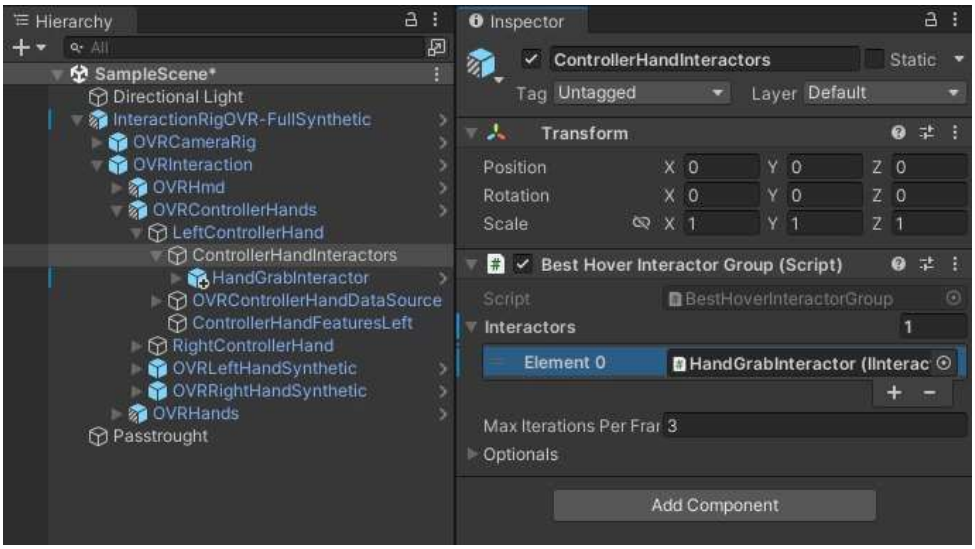


d. Membuat *Canvas* yang bisa disentuh

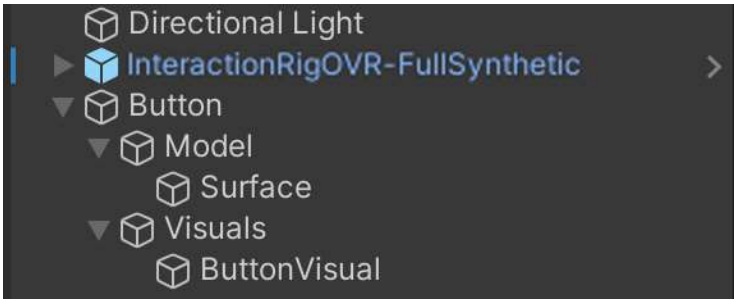
1. Pada bagian pencarian di tab *Project*, cari *HandPokeInteractor*. Pastikan filter pencarian diatur ke *All* atau *In Packages*, karena pengaturan *default* hanya mencari dalam *In Assets*.
2. Seret *prefab HandPokeInteractor* dari hasil pencarian ke dalam tab Hierarchy ke *InteractionRigOVR-FullSynthetic* > *OVRInteraction* > *OVRControllerHands* > *LeftControllerHands* > *ControllerHandInteractors*.



3. Tekan *LeftControllerHands* dan pada tab *Inspector*, terdapat komponen *Best Hover Interactor Group*, klik tanda + untuk menambahkan elemen ke daftar *Interactors*.
4. Seret *HandPokeInteractor* pada tab *Hierarchy* yang berada di bawah *LeftControllerHands* ke dalam kolom *Interactors* pada *Best Hover Interactor Group*.



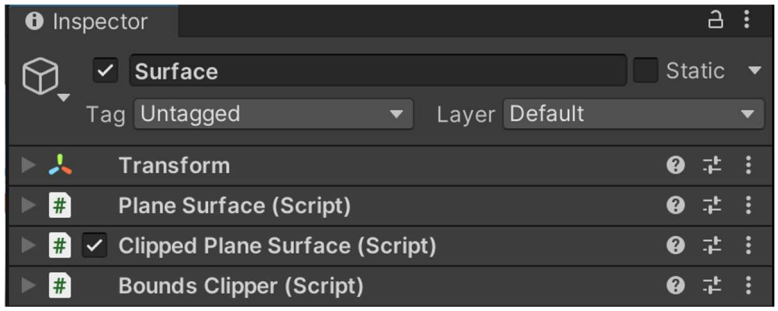
5. Lakukan langkah 2-4 untuk membuat interaksi *poke* pada *RightControllerHands*, juga pada *LeftHand*, dan *RightHand* yang berada di *OVRHands*.
6. Tambahkan *GameObject* kosong bernama *Button* ke *scene* dengan mengklik kanan di tab *Hierarchy* dan memilih *Create Empty*.
7. Posisikan *Button* di depan kamera.
8. Tambahkan dua *empty GameObject* sebagai *Child GameObject* ke Button bernama Model dan Visual dengan mengklik kanan *Button* lalu pilih *Create Empty*.
9. Tambahkan *empty GameObject* sebagai *Child GameObject* ke Model bernama *Surface*. *Surface* akan digunakan sebagai bagian belakang tombol.
10. Tambahkan sebuah *plane* ke Visual bernama *ButtonVisual* dengan mengklik kanan Visual dan kemudian pilih *3D Object* > *Plane*. Tab *Hierarchy* akan terlihat seperti ini.



11. Pada tab *Hierarchy*, pilih *Button* dan pada tab *Inspector*, tambahkan *Poke Interactable* dengan mengeklik *Add Component* lalu cari *Poke Interactable*.

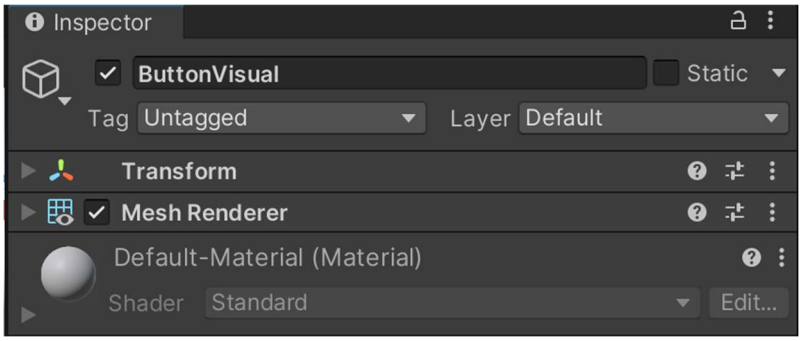
12. Pada, pilih *Surface* dan pada tab *Inspector*, tambahkan komponen berikut, sebagai permukaan tombol agar bisa ditekan:
    - *Plane Surface*
    - *Clipped Plane Surface*
    - *Bounds Clipper*

    GameObject Surface akan terlihat seperti ini.



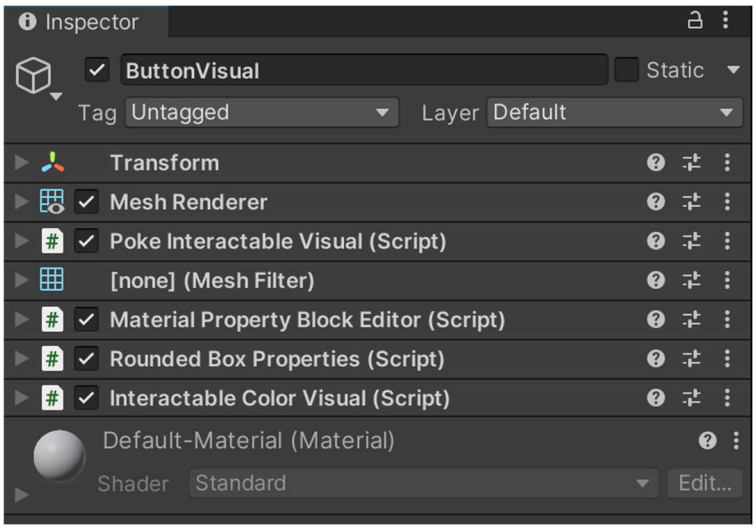
13. Pada tab *Hierarchy*, pilih *ButtonVisual* dan pada tab *Inspector*, hapus *Plane (Mesh Filter)* dan *Mesh Collider* dengan mengklik titik 3 pada setiap komponen, lalu pilih *Remove Component*.

    ButtonVisual setelah menghapus komponen akan terlihat seperti ini.

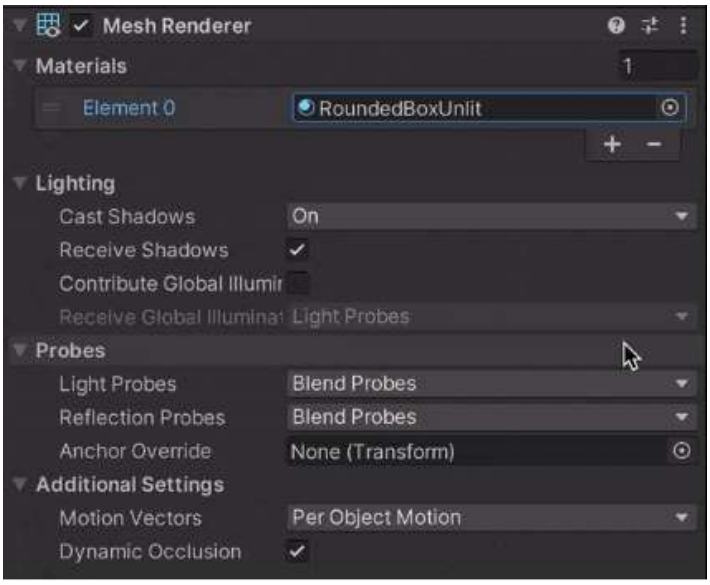


14. Tambahkan komponen-komponen berikut yang berguna untuk menentukan tampilan tombol:
    - Poke Interactable Visual
    - Mesh Filter
    - Material Property Block Editor
    - Rounded Box Properties
    - Interactable Color Visual

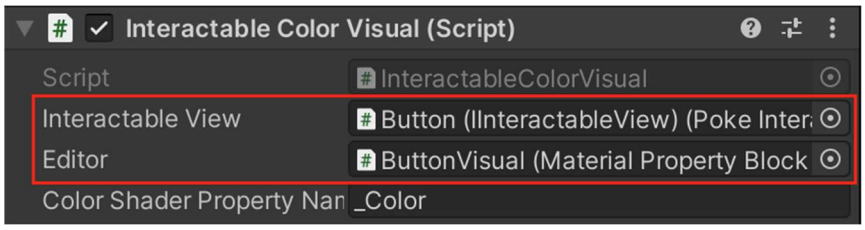
    ButtonVisual setelah ditambahkan komponen baru akan terlihat seperti ini.



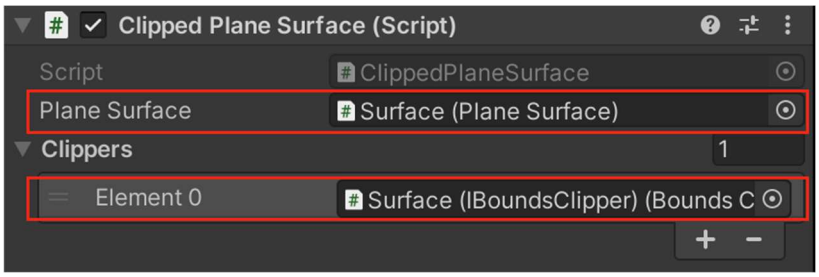
15. Pada komponen *Mesh Renderer*, dalam daftar Material, atur properti *Element* 0 ke *RoundedBoxUnlit* dengan mengklik tombol bulat kecil di sebelah kanan bidang *input* dan mencari *RoundedBoxUnlit*. Jika material tidak muncul di hasil pencarian, maka pada kolom pencarian tab *Project*, masukkan *RoundedBoxUnlit*.



16. Pada komponen *Poke Interactable Visual*, atur properti *Poke Interactable* ke *Button* dan properti *Button Base Transform* ke *Surface*.
17. Pada komponen *Mesh Filter*, atur properti *Mesh* ke *Quad* dengan mengklik tombol *Object Picker* (tombol bulat kecil di sebelah kanan bidang input) dan mencari *Quad*. *Quad* membuat tombol menjadi persegi panjang.
18. Pada komponen *Interactable Color Visual*, atur properti *Interactable View* ke *Button* dan properti *Editor* ke *ButtonVisual*.



19. Pada tab *Hierarchy*, pilih *Surface* dan pada tab *Inspector*, di komponen *Transform*, di properti *Scale*, tetapkan Z ke 0,001.
20. Pada komponen *Clipped Plane Surface*, atur properti *Plane Surface* ke *Surface*.
21. Pada komponen yang sama, klik tanda + untuk menambahkan elemen ke daftar *Clippers*.
22. Tetapkan elemen ke *Surface*. Komponen *Clipped Plane Surface* akan terlihat seperti berikut.



23. Pada tab *Hierarchy*, pilih *Button* dan pada tab *Inspector*, di komponen *Poke Interactable*, atur properti *Surface Patch* ke *Surface*.
24. Pada tab *Hierarchy*, pilih *Visual*. Kemudian pindahkan *Visual* pada sumbu Z sehingga sedikit lebih dekat ke kamera daripada Model *GameObject*. Hal ini memungkinkan tombol bergerak mundur secara visual saat dipencet.

Lampiran 3 *Script* Unity

a. PauseManager.cs

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine.UI;
4. using UnityEngine;
5. using UnityEngine.SceneManagement;
6. 
7. public class PauseManager : MonoBehaviour
8. {
9.     public static bool GameisPaused = false;
10.    public GameObject pauseMenuUI;
11.    public Button pauseButton;
12.    public Button resumeButton;
13.    public Button restartButton;
14.    public Button nextSceneButton;
15.    public Button mainMenuButton;
16.
17.    public string nextSceneMenuName = "Surgery";
18.    public string mainMenuSceneName = "Test";
19.
20.    public List<Button> buttonsToDisableOnPause;
21.
22.    void Start()
23.    {
24.        if (pauseButton != null)
25.            pauseButton.onClick.AddListener(PauseGame);
26.
27.        if (resumeButton != null)
28.            resumeButton.onClick.AddListener(ResumeGame);
29.
30.        if (restartButton != null)
31.            restartButton.onClick.AddListener(RestartGame);
32.
```

```
33.         if (nextSceneButton != null)
34.             nextSceneButton.onClick.AddListener(LoadNextScene)
    ;
35.
36.         if (mainMenuButton != null)
37.             mainMenuButton.onClick.AddListener(LoadMainMenu);
38.
39.         Time.timeScale = 1f;
40.         GameisPaused = false;
41.     }
42.
43.     void PauseGame()
44.     {
45.         pauseMenuUI.SetActive(true);
46.         Time.timeScale = 0f;
47.         GameisPaused = true;
48.
49.         foreach (Button button in buttonsToDisableOnPause)
50.         {
51.             button.interactable = false;
52.         }
53.     }
54.
55.     void ResumeGame()
56.     {
57.         pauseMenuUI.SetActive(false);
58.         Time.timeScale = 1f;
59.         GameisPaused = false;
60.
61.         foreach (Button button in buttonsToDisableOnPause)
62.         {
63.             button.interactable = true;
64.         }
65.     }
66.
67.     void RestartGame()
```

```
68.     {
69.         SceneManager.LoadScene(SceneManager.GetActiveScene().b
   uildIndex);
70.     }
71.
72.     void LoadNextScene()
73.     {
74.         SceneManager.LoadScene(nextSceneMenuName);
75.     }
76.
77.     void LoadMainMenu()
78.     {
79.         SceneManager.LoadScene(mainMenuSceneName);
80.     }
81.}
```

Rentang Kode	**Fungsi/Segmen Kode**	**Deskripsi**
1-6	Deklarasi *library*	Direktif menggunakan untuk pustaka yang diperlukan
7-8	Deklarasi Kelas	Deklarasi kelas "*PauseManager*"
9-21	Deklarasi Variabel	Variabel publik dan privat untuk kelas
22-42	*Start*()	Mengatur pendengar klik tombol dan menginisialisasi status permainan
43-54	*PauseGame*()	Menghentikan permainan dan menonaktifkan tombol yang ditentukan
55-66	*ResumeGame*()	Melanjutkan permainan dan mengaktifkan tombol yang ditentukan
67-71	*RestartGame*()	Memulai ulang *scene* saat ini
72-76	*LoadNextScene*()	Memuat *scene* berikutnya
77-81	*LoadMainMenu*()	Memuat *scene* menu utama

b. ActivateObject.cs

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4.
5. public class ActivateObject : MonoBehaviour
6. {
7.     public List<GameObject> ActiveGameObjects;
8.
9.     public void activate()
10.    {
11.        foreach (var activeGameObjects in ActiveGameObjects)
12.            if (activeGameObjects.activeInHierarchy == false)
13.                activeGameObjects.SetActive(true);
14.            else
15.                activeGameObjects.SetActive(true);
16.    }
17.}
```

Rentang Kode	Fungsi/Segmen Kode	Deskripsi
1-4	Deklarasi *library*	Direktif menggunakan untuk pustaka yang diperlukan
5-6	Deklarasi Kelas	Deklarasi kelas "*ActivateObject*"
7-8	Deklarasi Variabel	Variabel publik *ActiveGameObjects*
10-15	*activate*()	Metode untuk mengaktifkan objek-objek dalam daftar jika tidak aktif

c. DeactiveObject.cs

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4.
5. public class DeactivateObject : MonoBehaviour
```

```
6. {
7.     public List<GameObject> DeactiveGameObjects;
8.
9.     public void deactivate()
10.    {
11.        foreach (var deactiveGameObjects in
   DeactiveGameObjects)
12.            if (deactiveGameObjects.activeInHierarchy == true)
13.                deactiveGameObjects.SetActive(false);
14.            else
15.                deactiveGameObjects.SetActive(false);
16.    }
17.}
```

Rentang Kode	**Fungsi/Segmen Kode**	**Deskripsi**
1-4	Deklarasi *library*	Direktif menggunakan untuk pustaka yang diperlukan
5-6	Deklarasi Kelas	Deklarasi kelas "*DeactivateObject*"
7-8	Deklarasi Variabel	Variabel publik *DeactiveGameObject*
10-15	*deactivate*()	Metode untuk menonaktifkan objek-objek dalam daftar jika aktif

d. GrabActiveDeactive.cs

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4. using Oculus.Interaction;
5.
6. public class GrabActiveDeactive : MonoBehaviour
7. {
8.     public OneGrabFreeTransformerEdited transformer;
9.     public List<GameObject> ActiveGameObjects;
10.    public List<GameObject> DeactiveGameObjects;
```

```
11.     public List<GameObject> DestroyGameObjects;
12.
13.     void Update()
14.     {
15.         if (transformer.IsTransforming)
16.         {
17.             activate();
18.             deactivate();
19.             destroy();
20.         }
21.     }
22.
23.     public void activate()
24.     {
25.         foreach (var activeGameObjects in ActiveGameObjects)
26.             if (activeGameObjects.activeInHierarchy == false)
27.                 activeGameObjects.SetActive(true);
28.             else
29.                 activeGameObjects.SetActive(true);
30.     }
31.
32.     public void deactivate()
33.     {
34.         foreach (var deactiveGameObjects in
    DeactiveGameObjects)
35.             if (deactiveGameObjects.activeInHierarchy == true)
36.                 deactiveGameObjects.SetActive(false);
37.             else
38.                 deactiveGameObjects.SetActive(false);
39.     }
40.
41.     public void destroy()
42.     {
43.         foreach (var gameObjectToDestroy in
    DestroyGameObjects)
44.         {
```

```
45.             Destroy(gameObjectToDestroy);
46.         }
47.         DestroyGameObjects.Clear();
48.     }
49.}
```

Rentang Kode	Fungsi/Segmen Kode	Deskripsi
1-5	Deklarasi *library*	Direktif menggunakan untuk pustaka yang diperlukan
6-7	Deklarasi Kelas	Deklarasi kelas *GrabActiveDeactive*
8-12	Deklarasi Variabel	Deklarasi variabel publik *transformer*, *ActiveGameObjects*, *DeactiveGameObjects*, dan *DestroyGameObjects*
13-22	*Update*()	Memeriksa apakah *transformer* sedang mentransformasi, lalu memanggil metode *activate*(), *deactivate*(), dan *destroy*()
23-31	*activate*()	Metode untuk mengaktifkan objek-objek dalam daftar jika tidak aktif
32-40	*deactivate*()	Metode untuk menonaktifkan objek-objek dalam daftar jika aktif
41-49	*destroy*()	Metode untuk menghancurkan objek-objek dalam daftar dan mengosongkan daftar tersebut

e. ProgresActive.cs

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using TMPro;
4. using UnityEngine.UI;
5. using UnityEngine;
6.
7. public class ProgresActive : MonoBehaviour
```

```
8. {
9.     [Header("Text")]
10.    public TMP_Text progress;
11.    [Header("List")]
12.    public List<GameObject> checkObject;
13.    public List<GameObject> activeObject;
14.    public List<GameObject> deactiveObject;
15.    private void Update()
16.    {
17.        CheckAndDisplayAccuracy();
18.        UpdateObjectsBasedOnAccuracy();
19.    }
20.    public void CheckAndDisplayAccuracy()
21.    {
22.        int activeCount = CountActiveObjects();
23.        float totalObjects = checkObject.Count;
24.        float progressPercentage = 100f * (1 - (activeCount /
   totalObjects));
25.        progress.text = $"Progress: {progressPercentage:F2}%";
26.    }
27.
28.    int CountActiveObjects()
29.    {
30.        int count = 0;
31.        foreach (GameObject obj in checkObject)
32.        {
33.            if (obj.activeSelf)
34.            {
35.                count++;
36.            }
37.        }
38.        return count;
39.    }
40.
41.    void UpdateObjectsBasedOnAccuracy()
42.    {
```

```
43.         int activeCount = CountActiveObjects();
44.         float totalObjects = checkObject.Count;
45.         float progressPercentage = 100f * (1 - (activeCount /
    totalObjects));
46.         if (progressPercentage >= 100f)
47.         {
48.             ActivateObjects(activeObject);
49.             DeactivateObjects(deactiveObject);
50.         }
51.     }
52.
53.     void ActivateObjects(List<GameObject> objectList)
54.     {
55.         foreach (GameObject obj in objectList)
56.         {
57.             obj.SetActive(true);
58.         }
59.     }
60.
61.     void DeactivateObjects(List<GameObject> objectList)
62.     {
63.         foreach (GameObject obj in objectList)
64.         {
65.             obj.SetActive(false);
66.         }
67.     }
68. }
```

Rentang Kode	**Fungsi/Segmen Kode**	**Deskripsi**
1-6	Deklarasi *library*	Direktif menggunakan untuk pustaka yang diperlukan
7-8	Deklarasi Kelas	Deklarasi kelas "*ProgresActive*"

Rentang Kode	Fungsi/Segmen Kode	Deskripsi
9-15	Deklarasi Variabel	Deklarasi variabel publik *progress*, *checkObject*, *activeObject*, dan *deactiveObject*
16-20	*Update*()	Memanggil metode *CheckAndDisplayAccuracy*() dan *UpdateObjectsBasedOnAccuracy*() setiap *frame*
21-27	*CheckAndDisplayAccuracy*()	Menghitung objek yang aktif, menghitung persentase *progress*, dan menampilkan *progress*
29-38	*CountActiveObjects*()	Menghitung dan mengembalikan jumlah objek yang aktif dalam *checkObject*
40-50	*UpdateObjectsBasedOnAccuracy*()	Memeriksa persentase *progress* dan mengaktifkan atau menonaktifkan objek berdasarkan *progress*
52-56	*ActivateObjects*(*List<GameObject> objectList*)	Mengaktifkan semua objek dalam daftar *objectList*
58-62	*DeactivateObjects*(*List<GameObject> objectList*)	Menonaktifkan semua objek dalam daftar *objectList*

f. SingleModeMeasurement.cs

```
1. using System.Collections;
2. using System.Collections;
3. using UnityEngine;
4. using TMPro;
5. using Oculus.Interaction;
6.
7. public class SingleModeMeasurement : MonoBehaviour
8. {
```

```
9.      [Header("Ruler Object")]
10.     public GameObject tip;
11.     public OneGrabFreeTransformerEdited transformer;
12.
13.     [Header("Points")]
14.     public GameObject pointA;
15.     public GameObject pointB;
16.     public RaycastObject raycastObjectScript;
17.
18.     float distance;
19.     string exportString;
20.
21.     [Header("Text")]
22.     public GameObject distanceCanvas;
23.     public TMP_Text textField;
24.
25.     [Header("Line Renderers")]
26.     public GameObject lineRenderer;
27.     public LineRenderer lineRendererChild;
28.
29.     [Header("Haptic Feedback")]
30.     public float vibrationIntensity = 0.5f;
31.     public float vibrationDuration = 0.1f;
32.     public AudioSource soundEffect;
33.
34.     private int buttonPressCount = 0;
35.
36.     void Start()
37.     {
38.         pointA.transform.position = Vector3.zero;
39.         pointB.transform.position = Vector3.zero;
40.         ResetPointsAndLine();
41.         distanceCanvas.SetActive(false);
42.     }
43.
44.     void Update()
```

```
45.     {
46.         if (pointA.activeSelf && pointB.activeSelf)
47.         {
48.             UpdateDistanceAndCanvas();
49.             Measure();
50.         }
51.
52.         if (!raycastObjectScript.IsRaycastActive() &&
    transformer.IsTransforming &&
    OVRInput.GetDown(OVRInput.Button.One,
    OVRInput.Controller.RTouch))
53.         {
54.             PlacePoint();
55.             soundEffect.Play();
56.         }
57.     }
58.
59.     private void UpdateDistanceAndCanvas()
60.     {
61.         float distance =
    Vector3.Distance(pointA.transform.position,
    pointB.transform.position);
62.
63.         lineRendererChild.SetPosition(0,
    pointA.transform.position);
64.         lineRendererChild.SetPosition(1,
    pointB.transform.position);
65.
66.         Vector3 midPoint = (pointA.transform.position +
    pointB.transform.position) / 2;
67.         distanceCanvas.transform.position = midPoint;
68.         distance *= 1000;
69.         textField.text = distance.ToString("N2") + "mm";
70.     }
71.
72.     void Measure()
```

```
73.     {
74.         float distance =
    Vector3.Distance(pointA.transform.position,
    pointB.transform.position);
75.         distance *= 1000;
76.         textField.text = distance.ToString("N2") + "mm";
77.     }
78.
79.     public void HandlePointPlacement(Vector3 position, bool
    hitDetected)
80.     {
81.         if (buttonPressCount == 0)
82.         {
83.             PlacePointRaycast(pointA, position, hitDetected);
84.             buttonPressCount++;
85.         }
86.         else if (buttonPressCount == 1)
87.         {
88.             PlacePointRaycast(pointB, position, hitDetected);
89.             lineRenderer.SetActive(true);
90.             Vector3 midPoint = (pointA.transform.position +
    pointB.transform.position) / 2;
91.             distanceCanvas.transform.position = midPoint;
92.             distanceCanvas.SetActive(true);
93.             buttonPressCount = 0;
94.         }
95.     }
96.
97.     private void PlacePointRaycast(GameObject point, Vector3
    position, bool hitDetected)
98.     {
99.         point.SetActive(true);
100.             point.transform.position = position;
101.
102.             StartCoroutine(VibrateController(vibrationDurati
    on, vibrationIntensity));
```

```
103.         }
104.
105.         void PlacePoint()
106.         {
107.             Vector3 tipPosition = tip.transform.position;
108.             if (buttonPressCount == 0)
109.             {
110.                 pointA.SetActive(true);
111.                 pointA.transform.position = tipPosition;
112.                 buttonPressCount++;
113.             }
114.             else if (buttonPressCount == 1)
115.             {
116.                 pointB.SetActive(true);
117.                 pointB.transform.position = tipPosition;
118.                 lineRenderer.SetActive(true);
119.                 Vector3 midPoint =
    (pointA.transform.position + pointB.transform.position) / 2;
120.                 distanceCanvas.transform.position =
    midPoint;
121.                 distanceCanvas.SetActive(true);
122.                 buttonPressCount = 0;
123.             }
124.             StartCoroutine(VibrateController(vibrationDurati
    on, vibrationIntensity));
125.         }
126.
127.         private IEnumerator VibrateController(float
    duration, float intensity)
128.         {
129.             OVRInput.SetControllerVibration(intensity,
    intensity, OVRInput.Controller.RTouch);
130.             yield return new WaitForSeconds(duration);
131.             OVRInput.SetControllerVibration(0, 0,
    OVRInput.Controller.RTouch);
132.         }
```

```
133.         public void ResetPointsAndLine()
134.         {
135.             pointA.transform.position = Vector3.zero;
136.             pointB.transform.position = Vector3.zero;
137.             pointA.SetActive(false);
138.             pointB.SetActive(false);
139.             lineRenderer.SetActive(false);
140.             distanceCanvas.SetActive(false);
141.             buttonPressCount = 0;
142.         }
143.     }
```

Rentang Kode	**Fungsi/Segmen Kode**	**Deskripsi**
1-5	Deklarasi *library*	Direktif menggunakan untuk pustaka yang diperlukan
6-7	Deklarasi Kelas	Deklarasi kelas *SingleModeMeasurement*
8-31	Deklarasi Variabel	Deklarasi variabel publik dan privat
35-41	*Start*()	Inisialisasi posisi *pointA* dan *pointB*, menyembunyikan objek dan *canvas*, serta me-*reset* garis dan titik
43-56	*Update*()	Memperbarui jarak dan *canvas*, mengukur, serta menangani penempatan titik berdasarkan *input*
58-69	*UpdateDistanceAndCanvas*()	Menghitung jarak antara *pointA* dan *pointB*, memperbarui posisi *line renderer*, dan memperbarui teks jarak pada *canvas*
71-76	*Measure*()	Menghitung dan memperbarui teks jarak antara *pointA* dan *pointB*
78-94	*HandlePointPlacement*(*Vector3, bool*)	Menangani penempatan titik menggunakan *raycast* dan mengaktifkan *line renderer* serta *canvas*

Rentang Kode	Fungsi/Segmen Kode	Deskripsi
96-102	*PlacePointRaycast* (*Vector3, bool*)	Menempatkan titik menggunakan *raycast* dan memulai getaran kontroler
104-124	*PlacePoint*()	Menempatkan titik berdasarkan posisi *tip*, mengaktifkan *line renderer* dan *canvas*, serta memulai getaran kontroler
126-131	*VibrateController* (*float, float*)	Menangani getaran kontroler untuk durasi dan intensitas tertentu
133-142	*ResetPointsAnd Line*()	Me-*reset* posisi dan status aktif *pointA*, *pointB*, *line renderer*, dan *canvas*, serta mengatur ulang jumlah penekanan tombol

g. MultiModeMeasurement.cs

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4. using Oculus.Interaction;
5. using TMPro;
6.
7. public class MultiModeMeasurement : MonoBehaviour
8. {
9.     [Header("Ruler Object")]
10.    public GameObject tip;
11.    public OneGrabFreeTransformerEdited transformer;
12.
13.    [Header("Points")]
14.    public GameObject measurePoint;
15.    public GameObject thisObject;
16.    public RaycastObject raycastObjectScript;
17.
18.    [Header("Text")]
19.    public GameObject distanceTextPrefab;
20.    public TMP_Text textField;
```

```
21.
22.     float totaldDistance;
23.     float totalarea;
24.
25.     List<GameObject> refPoints;
26.     List<GameObject> refLines;
27.     List<GameObject> distanceDisplays;
28.
29.     [Header("Line Renderers")]
30.     public GameObject lineRenderer;
31.     public bool closed = false;
32.
33.     [Header("Haptic Feedback")]
34.     public float vibrationIntensity = 0.5f;
35.     public float vibrationDuration = 0.1f;
36.     public AudioSource soundEffect;
37.
38.     void Start()
39.     {
40.         refPoints = new List<GameObject>();
41.         refLines = new List<GameObject>();
42.         distanceDisplays = new List<GameObject>();
43.         if (thisObject.activeSelf)
44.         {
45.             Debug.Log("Is Active");
46.         }
47.         measurePoint.SetActive(false);
48.         lineRenderer.SetActive(false);
49.         distanceTextPrefab.SetActive(false);
50.     }
51.
52.     void Update()
53.     {
54.         if (!raycastObjectScript.IsRaycastActive() &&
   transformer.IsTransforming &&
```

```
   OVRInput.GetDown(OVRInput.Button.One,
   OVRInput.Controller.RTouch))
55.        {
56.            CreateMeasurePointAndLine();
57.            StartCoroutine(VibrateController(vibrationDuration
   , vibrationIntensity));
58.            soundEffect.Play();
59.        }
60.        UpdateLinesAndDistanceTexts();
61.    }
62.
63.    private void UpdateLinesAndDistanceTexts()
64.    {
65.        for (int i = 0; i < refPoints.Count - 1; i++)
66.        {
67.            LineRenderer lineRenderer =
   refLines[i].GetComponent<LineRenderer>();
68.            if (lineRenderer != null)
69.            {
70.                lineRenderer.SetPosition(0,
   refPoints[i].transform.position);
71.                lineRenderer.SetPosition(1, refPoints[i +
   1].transform.position);
72.            }
73.
74.            float distance =
   Vector3.Distance(refPoints[i].transform.position, refPoints[i
   + 1].transform.position);
75.            Vector3 midPoint =
   (refPoints[i].transform.position + refPoints[i +
   1].transform.position) / 2;
76.            distanceDisplays[i].transform.position = midPoint;
77.            TMP_Text distanceText =
   distanceDisplays[i].GetComponentInChildren<TMP_Text>();
78.            if (distanceText != null)
79.            {
```

```
80.                 distance *= 1000;
81.                 distanceText.text = distance.ToString("N2") +
    "mm";
82.             }
83.         }
84.     }
85.
86.     private void CreateMeasurePointAndLine()
87.     {
88.         GameObject newPoint = Instantiate(measurePoint,
    tip.transform.position, Quaternion.identity);
89.         newPoint.SetActive(true);
90.         newPoint.transform.parent = thisObject.transform;
91.         refPoints.Add(newPoint);
92.
93.         if (refPoints.Count > 1)
94.         {
95.             GameObject newLine = Instantiate(lineRenderer,
    tip.transform.position, Quaternion.identity);
96.             newLine.SetActive(true);
97.             newLine.transform.parent = thisObject.transform;
98.             refLines.Add(newLine);
99.
100.                GameObject newTextDisplay =
    Instantiate(distanceTextPrefab, Vector3.zero,
    Quaternion.identity);
101.                newTextDisplay.SetActive(true);
102.                distanceDisplays.Add(newTextDisplay);
103.            }
104.        }
105.
106.        public void PlaceMeasurePoint(Vector3 position, bool
    hitDetected)
107.        {
108.            if (hitDetected)
109.            {
```

```
110.                 GameObject newPoint =
   Instantiate(measurePoint, position, Quaternion.identity);
111.                 newPoint.SetActive(true);
112.                 newPoint.transform.parent =
   thisObject.transform;
113.                 refPoints.Add(newPoint);
114.             }
115.             if (refPoints.Count > 1)
116.             {
117.                 GameObject newLine =
   Instantiate(lineRenderer, tip.transform.position,
   Quaternion.identity);
118.                 newLine.SetActive(true);
119.                 newLine.transform.parent =
   thisObject.transform;
120.                 refLines.Add(newLine);
121.
122.                 GameObject newTextDisplay =
   Instantiate(distanceTextPrefab, Vector3.zero,
   Quaternion.identity);
123.                 newTextDisplay.SetActive(true);
124.                 distanceDisplays.Add(newTextDisplay);
125.             }
126.             StartCoroutine(VibrateController(vibrationDurati
   on, vibrationIntensity));
127.         }
128.
129.         private IEnumerator VibrateController(float
   duration, float intensity)
130.         {
131.             OVRInput.SetControllerVibration(intensity,
   intensity, OVRInput.Controller.RTouch);
132.             yield return new WaitForSeconds(duration);
133.             OVRInput.SetControllerVibration(0, 0,
   OVRInput.Controller.RTouch);
134.         }
```

```
135.
136.         public void resetpoints()
137.         {
138.             foreach (var item in refLines)
139.             {
140.                 if (item != null)
141.                     Destroy(item);
142.             }
143.             refLines.Clear();
144.
145.             foreach (var item in refPoints)
146.             {
147.                 if (item != null)
148.                     Destroy(item);
149.             }
150.             refPoints.Clear();
151.             foreach (var display in distanceDisplays)
152.             {
153.                 if (display != null)
154.                     Destroy(display);
155.             }
156.             distanceDisplays.Clear();
157.         }
158.     }
```

Rentang Kode	**Fungsi/Segmen Kode**	**Deskripsi**
1-5	Deklarasi *library*	Direktif menggunakan untuk pustaka yang diperlukan
6-7	Deklarasi Kelas	Deklarasi kelas *MultiModeMeasurement*
8-36	Deklarasi Variabel	Deklarasi variabel publik dan privat
38-50	*Start*()	Inisialisasi daftar objek, pengecekan status aktif, dan menyembunyikan objek serta kanvas

Rentang Kode	Fungsi/Segmen Kode	Deskripsi
52-61	*Update*()	Memeriksa kondisi *raycast* dan transformasi untuk membuat titik dan garis pengukuran serta memperbarui garis dan Serta kanvas
63-84	*UpdateLinesAnd DistanceTexts* ()	Memperbarui posisi garis dan teks jarak berdasarkan posisi titik pengukuran
86-104	*CreateMeasure PointAndLine*()	Membuat titik dan garis pengukuran baru serta menambahkan ke daftar objek referensi
106-127	*PlaceMeasurePoint* (*Vector3, bool*)	Menempatkan titik pengukuran berdasarkan posisi *raycast* dan menambahkan garis serta teks jarak jika ada lebih dari satu titik
129-134	*VibrateController* (*float, float*)	Menangani getaran kontroler untuk durasi dan intensitas tertentu
136-157	*resetpoints*()	Menghapus semua garis, titik pengukuran, dan tampilan jarak dari daftar serta menghancurkan objek terkait

h. AngleModeMeasurement.cs

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4. using Oculus.Interaction;
5. using TMPro;
6.
7. public class AngleModeMeasurement : MonoBehaviour
8. {
9.     [Header("Ruler Object")]
10.    public GameObject tip;
11.    public OneGrabFreeTransformerEdited transformer;
```

```
12.
13.     [Header("Points")]
14.     public GameObject pointA;
15.     public GameObject pointB;
16.     public GameObject pointC;
17.     public RaycastObject raycastObjectScript;
18.
19.     [Header("Text")]
20.     public GameObject distanceCanvas;
21.     public TMP_Text textField;
22.
23.     [Header("Line Renderers")]
24.     public GameObject lineRendererAB;
25.     public GameObject lineRendererBC;
26.     public GameObject lineRendererAC;
27.
28.     [Header("Haptic Feedback")]
29.     public float vibrationIntensity = 0.5f;
30.     public float vibrationDuration = 0.1f;
31.     public AudioSource soundEffect;
32.
33.     string exportString;
34.     private int buttonPressCount = 0;
35.
36.     void Start()
37.     {
38.         pointA.SetActive(false);
39.         pointB.SetActive(false);
40.         pointC.SetActive(false);
41.         distanceCanvas.SetActive(false);
42.         lineRendererAB.SetActive(false);
43.         lineRendererBC.SetActive(false);
44.         lineRendererAC.SetActive(false);
45.     }
46.
47.     void Update()
```

```
48.     {
49.         if (!raycastObjectScript.IsRaycastActive() &&
    transformer.IsTransforming &&
    OVRInput.GetDown(OVRInput.Button.One,
    OVRInput.Controller.RTouch))
50.         {
51.             PlacePoint();
52.             soundEffect.Play();
53.         }
54.
55.         if (pointA.activeSelf && pointB.activeSelf &&
    pointC.activeSelf)
56.         {
57.             UpdateDistanceCanvasPosition();
58.             Measure();
59.         }
60.     }
61.
62.     void UpdateDistanceCanvasPosition()
63.     {
64.         Vector3 centroid = (pointA.transform.position +
    pointB.transform.position + pointC.transform.position) / 3;
65.         distanceCanvas.transform.position = centroid;
66.     }
67.
68.     void Measure()
69.     {
70.         float AngleR = Vector3.Angle(pointA.transform.position
    - pointB.transform.position, pointC.transform.position -
    pointB.transform.position);
71.         float AngleG = Vector3.Angle(pointB.transform.position
    - pointC.transform.position, pointA.transform.position -
    pointC.transform.position);
72.         float AngleB = Vector3.Angle(pointB.transform.position
    - pointA.transform.position, pointC.transform.position -
    pointA.transform.position);
```

```
73.         textField.text = "R, G, B = " + AngleR.ToString("N0")
    + "°, " + AngleG.ToString("N0") + "°, " +
    AngleB.ToString("N0") + "°";
74.     }
75.
76.     public void PlaceAnglePoint(Vector3 position, bool
    hitDetected)
77.     {
78.         if (hitDetected)
79.         {
80.             GameObject selectedPoint = SelectPointToPlace();
81.             if (selectedPoint != null)
82.             {
83.                 selectedPoint.transform.position = position;
84.                 selectedPoint.SetActive(true);
85.                 HandleLineRenderers();
86.                 StartCoroutine(VibrateController(vibrationDura
    tion, vibrationIntensity));
87.             }
88.
89.             buttonPressCount++;
90.             if (buttonPressCount > 2)
91.             {
92.                 buttonPressCount = 0;
93.             }
94.         }
95.     }
96.
97.     private GameObject SelectPointToPlace()
98.     {
99.         switch (buttonPressCount)
100.             {
101.                 case 0: return pointA;
102.                 case 1: return pointB;
103.                 case 2: return pointC;
104.                 default: return null;
```

```
105.             }
106.         }
107.
108.         private void HandleLineRenderers()
109.         {
110.             switch (buttonPressCount)
111.             {
112.                 case 0:
113.                     break;
114.                 case 1:
115.                     lineRendererAB.SetActive(true);
116.                     break;
117.                 case 2:
118.                     lineRendererBC.SetActive(true);
119.                     lineRendererAC.SetActive(true);
120.                     distanceCanvas.SetActive(true);
121.                     break;
122.             }
123.         }
124.
125.         void PlacePoint()
126.         {
127.             switch (buttonPressCount)
128.             {
129.                 case 0:
130.                     pointA.transform.position =
    tip.transform.position;
131.                     pointA.SetActive(true);
132.                     StartCoroutine(VibrateController(vibrati
    onDuration, vibrationIntensity));
133.                     break;
134.                 case 1:
135.                     pointB.transform.position =
    tip.transform.position;
136.                     pointB.SetActive(true);
137.                     lineRendererAB.SetActive(true);
```

```
138.                     StartCoroutine(VibrateController(vibrati
    onDuration, vibrationIntensity));
139.                     break;
140.                 case 2:
141.                     pointC.transform.position =
    tip.transform.position;
142.                     pointC.SetActive(true);
143.                     lineRendererBC.SetActive(true);
144.                     lineRendererAC.SetActive(true);
145.                     distanceCanvas.SetActive(true);
146.                     StartCoroutine(VibrateController(vibrati
    onDuration, vibrationIntensity));
147.                     break;
148.             }
149.             buttonPressCount++;
150.             if (buttonPressCount > 2)
151.             {
152.                 buttonPressCount = 0;
153.             }
154.         }
155.
156.         private IEnumerator VibrateController(float
    duration, float intensity)
157.         {
158.             OVRInput.SetControllerVibration(intensity,
    intensity, OVRInput.Controller.RTouch);
159.             yield return new WaitForSeconds(duration);
160.             OVRInput.SetControllerVibration(0, 0,
    OVRInput.Controller.RTouch);
161.         }
162.
163.         public void ResetPoints()
164.         {
165.             pointA.transform.position = Vector3.zero;
166.             pointB.transform.position = Vector3.zero;
167.             pointC.transform.position = Vector3.zero;
```

```
168.             pointA.SetActive(false);
169.             pointB.SetActive(false);
170.             pointC.SetActive(false);
171.             lineRendererAB.SetActive(false);
172.             lineRendererBC.SetActive(false);
173.             lineRendererAC.SetActive(false);
174.             distanceCanvas.SetActive(false);
175.             buttonPressCount = 0;
176.         }
177.     }
```

Rentang Kode	**Fungsi/Segmen Kode**	**Deskripsi**
1-5	Deklarasi *library*	Direktif menggunakan untuk pustaka yang diperlukan
6-7	Deklarasi Kelas	Deklarasi kelas *AngleModeMeasurement*
8-34	Deklarasi Variabel	Deklarasi variabel publik dan privat
36-45	*Start*()	Inisialisasi status titik dan objek lain sebagai tidak aktif
47-60	*Update*()	Menangani *input* pengguna dan memperbarui posisi kanvas serta mengukur sudut
62-66	*UpdateDistance CanvasPosition*()	Memperbarui posisi kanvas jarak berdasarkan posisi titik
68-75	*Measure*()	Menghitung sudut antara titik dan memperbarui teks pada kanvas
76-95	*PlaceAnglePoint* (*Vector3, bool*)	Menempatkan titik sudut berdasarkan *input* dan memperbarui *line renderer* serta getaran kontroler
97-106	*SelectPointTo Place*()	Memilih titik yang akan ditempatkan berdasarkan jumlah penekanan tombol
108-123	*HandleLine Renderers*()	Menangani aktivasi *line renderer* berdasarkan jumlah penekanan tombol

Rentang Kode	Fungsi/Segmen Kode	Deskripsi
125-154	*PlacePoint*()	Menempatkan titik pada posisi ujung, memperbarui *line* renderer dan kanvas jarak, serta memulai getaran kontroler
156-161	*VibrateController* (*float, float*)	Menangani getaran kontroler untuk durasi dan intensitas tertentu
163-176	*ResetPoints*()	Mengatur ulang posisi titik dan status aktif titik serta *line renderer* dan kanvas jarak

i. MeasureModeController.cs

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4.
5. public class MeasurementModeController : MonoBehaviour
6. {
7.     public SingleModeMeasurement singleMode;
8.     public MultiModeMeasurement multiMode;
9.     public AngleModeMeasurement angleMode;
10.
11.    public void ActivateMultiMode()
12.    {
13.        singleMode.enabled = false;
14.        multiMode.enabled = true;
15.        angleMode.enabled = false;
16.    }
17.
18.    public void ActivateAngleMode()
19.    {
20.        singleMode.enabled = false;
21.        multiMode.enabled = false;
22.        angleMode.enabled = true;
23.    }
```

```
24.
25.     private void SwitchToSingleMode()
26.     {
27.         singleMode.enabled = true;
28.         multiMode.enabled = false;
29.         angleMode.enabled = false;
30.     }
31. }
```

Rentang Kode	Fungsi/Segmen Kode	Deskripsi
1-3	Deklarasi *library*	Direktif menggunakan untuk pustaka yang diperlukan
4-5	Deklarasi Kelas	Deklarasi kelas *MeasurementModeController*
7-9	Deklarasi Variabel	Deklarasi variabel publik untuk mode pengukuran
11-16	*ActivateMultiMode*()	Mengaktifkan mode pengukuran *multi mode* dan menonaktifkan mode lainnya
18-23	*ActivateAngleMode*()	Mengaktifkan mode pengukuran *angle mode* dan menonaktifkan mode lainnya
25-30	*SwitchToSingleMode*()	Mengaktifkan mode pengukuran *single mode* dan menonaktifkan mode lainnya

j. AirModeMarking.cs

```
1. using System.Collections.Generic;
2. using System.Collections;
3. using UnityEngine;
4. using UnityEngine.UI;
5. using Oculus.Interaction;
6.
7. public class AirModeMarking : MonoBehaviour
8. {
9.     public Transform laserOrigin;
```

```
10.     public Material lineMaterial;
11.     public OneGrabFreeTransformerEdited transformer;
12.
13.     public Slider widthSlider;
14.     [Range(0.001f, 0.025f)]
15.     public float lineWidth = 0.003f;
16.
17.     [Header("Haptic Feedback")]
18.     public float vibrationIntensity = 0.5f;
19.     private bool isVibrating = false;
20.     public GameObject markingSound;
21.     private List<LineRenderer> createdLines = new
    List<LineRenderer>();
22.     private List<LineRenderer> undoLines = new
    List<LineRenderer>();
23.     private int maxUndoSteps = 3;
24.     private bool isDrawing = false;
25.     private LineRenderer currentLineRenderer;
26.
27.
28.     void Start()
29.     {
30.         if (widthSlider != null)
31.         {
32.             widthSlider.onValueChanged.AddListener(SetLineWidt
    hFromSlider);
33.         }
34.     }
35.
36.     void Update()
37.     {
38.         if (transformer != null)
39.         {
40.             if (transformer.IsTransforming)
41.             {
```

```
42.                 if (OVRInput.Get(OVRInput.Button.One,
    OVRInput.Controller.RTouch))
43.                 {
44.                     if (!isDrawing)
45.                     {
46.                         StartDrawing();
47.                         markingSound.SetActive(true);
48.                     }
49.                     UpdateLine();
50.
51.                     if (!isVibrating)
52.                     {
53.                         StartCoroutine(VibrateController(0.2f,
    vibrationIntensity));
54.                     }
55.                 }
56.                 else if (isDrawing)
57.                 {
58.                     StopDrawing();
59.                     markingSound.SetActive(false);
60.                 }
61.             }
62.             else
63.             {
64.                 if (isDrawing)
65.                 {
66.                     StopDrawing();
67.                     markingSound.SetActive(false);
68.                 }
69.             }
70.         }
71.     }
72.
73.     public void SetLineWidthFromSlider(float value)
74.     {
75.         lineWidth = value;
```

```
76.         SetLineWidth(lineWidth);
77.     }
78.
79.     void StartDrawing()
80.     {
81.         isDrawing = true;
82.
83.         currentLineRenderer = CreateNewLineRenderer();
84.         currentLineRenderer.material = lineMaterial;
85.         SetLineWidth(lineWidth);
86.         UpdateLine();
87.     }
88.
89.     void UpdateLine()
90.     {
91.         if (currentLineRenderer != null)
92.         {
93.             currentLineRenderer.positionCount++;
94.             int index = currentLineRenderer.positionCount - 1;
95.             currentLineRenderer.SetPosition(index,
   laserOrigin.position);
96.         }
97.     }
98.
99.     void StopDrawing()
100.         {
101.             isDrawing = false;
102.             if (currentLineRenderer != null)
103.             {
104.                 createdLines.Add(currentLineRenderer);
105.                 undoLines.Insert(0, currentLineRenderer);
106.                 TrimUndoList();
107.             }
108.             currentLineRenderer = null;
109.         }
110.
```

```
111.         void TrimUndoList()
112.         {
113.             while (undoLines.Count > maxUndoSteps)
114.             {
115.                 undoLines.RemoveAt(undoLines.Count - 1);
116.             }
117.         }
118.
119.         public void SetLineWidth(float width)
120.         {
121.             if (currentLineRenderer != null)
122.             {
123.                 currentLineRenderer.widthMultiplier =
    Mathf.Max(width, 0.0f);
124.             }
125.         }
126.
127.         public void Undo()
128.         {
129.             if (createdLines.Count > 0)
130.             {
131.                 int lastIndex = createdLines.Count - 1;
132.                 Destroy(createdLines[lastIndex].gameObject);
133.                 createdLines.RemoveAt(lastIndex);
134.             }
135.         }
136.
137.         public void ResetAllLines()
138.         {
139.             foreach (var line in createdLines)
140.             {
141.                 Destroy(line.gameObject);
142.             }
143.             createdLines.Clear();
144.             undoLines.Clear();
145.         }
```

```
146.
147.         private IEnumerator VibrateController(float
   duration, float intensity)
148.         {
149.             OVRInput.SetControllerVibration(intensity,
   intensity, OVRInput.Controller.RTouch);
150.             yield return new WaitForSeconds(duration);
151.             OVRInput.SetControllerVibration(0, 0,
   OVRInput.Controller.RTouch);
152.         }
153.
154.         LineRenderer CreateNewLineRenderer()
155.         {
156.             GameObject newLineObject = new
   GameObject("LineRenderer");
157.             newLineObject.transform.SetParent(transform);
158.             LineRenderer lineRenderer =
   newLineObject.AddComponent<LineRenderer>();
159.             lineRenderer.material = lineMaterial;
160.             lineRenderer.positionCount = 0;
161.             return lineRenderer;
162.         }
163.     }
```

Rentang Kode	**Fungsi/Segmen Kode**	**Deskripsi**
1-5	Deklarasi *library*	Direktif menggunakan untuk pustaka yang diperlukan
6-7	Deklarasi Kelas	Deklarasi kelas *SingleModeMeasurement*
9-11	Deklarasi Variabel	Deklarasi variabel publik untuk laser, material *line renderer*, dan *transformer*
13-20	Deklarasi Variabel UI dan *Feedback*	Deklarasi variabel untuk *slider* lebar garis, *feedback* kontroler, dan suara penanda

21-25	Deklarasi Variabel Privat	Deklarasi daftar *line renderer* untuk garis yang dibuat dan *undo*, serta variabel status
28-34	*Start*()	Menginisialisasi *slider* lebar garis jika tidak *null*
36-71	*Update*()	Mengelola logika menggambar garis saat
Rentang Kode	**Fungsi/Segmen Kode**	**Deskripsi**
36-71	*Update*()	tombol ditekan dan *transformer* aktif
73-77	*SetLineWidthFrom Slider*(*float value*)	Mengatur lebar garis berdasarkan nilai *slider*
79-87	*StartDrawing*()	Memulai menggambar garis baru dan menginisialisasi *line renderer*
89-97	*UpdateLine*()	Memperbarui posisi garis saat menggambar
99-109	*StopDrawing*()	Menghentikan menggambar garis dan menambahkan garis yang dibuat ke daftar *undo*
111-117	*TrimUndoList*()	Memangkas daftar *undo* jika melebihi jumlah maksimum langkah *undo*
119-125	*SetLineWidth*(*float width*)	Mengatur lebar garis dari *line renderer* yang sedang aktif
127-135	*Undo*()	Menghapus garis terakhir yang dibuat dari daftar dan menghapusnya dari tampilan
137-145	*ResetAllLines*()	Menghapus semua garis yang dibuat dan mengosongkan daftar *undo*
147-152	*VibrateController* (*float duration, float intensity*)	Memberikan *feedback* getaran kontroler saat menggambar garis
154-162	*CreateNewLine Renderer()*	Membuat dan menginisialisasi objek *line renderer* baru

k. SnapModeMarking.cs

```
1. using System.Collections.Generic;
```

```
2. using System.Collections;
3. using UnityEngine;
4. using UnityEngine.UI;
5. using Oculus.Interaction;
6.
7. public class SnapModeMarking : MonoBehaviour
8. {
9.     [Header("Marker Object")]
10.    public Transform laserOrigin;
11.    public Material lineMaterial;
12.    public OneGrabFreeTransformerEdited transformer;
13.    public GameObject parentObject;
14.
15.    [Header("Thickness")]
16.    public Slider widthSlider;
17.    [Range(0.001f, 0.025f)]
18.    public float lineWidth = 0.003f;
19.
20.    private List<LineRenderer> createdLines = new
   List<LineRenderer>();
21.    private List<LineRenderer> undoLines = new
   List<LineRenderer>();
22.    private int maxUndoSteps = 3;
23.    private bool isDrawing = false;
24.    private LineRenderer currentLineRenderer;
25.    private List<GameObject> createdMeshObjects = new
   List<GameObject>();
26.
27.    [Header("Laser")]
28.    public GameObject laserObject;
29.    public LineRenderer laserLine;
30.    public Color defaultLaserColor = Color.red;
31.    public Color hitColor = Color.blue;
32.
33.    [Header("Haptic Feedback")]
34.    public GameObject markingSound;
```

```
35.     public float vibrationIntensity = 0.5f;
36.     private bool isVibrating = false;
37.
38.     void Start()
39.     {
40.         laserLine = laserObject.GetComponent<LineRenderer>();
41.         if (laserLine == null)
42.         {
43.             Debug.LogError("No LineRenderer component found on
    the laserObject GameObject.");
44.             return;
45.         }
46.         if (widthSlider != null)
47.         {
48.             widthSlider.onValueChanged.AddListener(SetLineWidt
    hFromSlider);
49.         }
50.         laserObject.SetActive(false);
51.     }
52.
53.     void Update()
54.     {
55.         if (transformer != null)
56.         {
57.
58.             if (transformer.IsTransforming)
59.             {
60.                 HandleLaser();
61.                 laserObject.SetActive(false);
62.
63.                 if (OVRInput.Get(OVRInput.Button.One,
    OVRInput.Controller.RTouch))
64.                 {
65.                     if (!isDrawing)
66.                     {
67.                         StartDrawing();
```

```
68.                         markingSound.SetActive(true);
69.                     }
70.                     UpdateLine();
71.
72.                     if (!isVibrating)
73.                     {
74.                         StartCoroutine(VibrateController(0.1f,
   vibrationIntensity));
75.                     }
76.                 }
77.                 else if (isDrawing)
78.                 {
79.                     StopDrawing();
80.                     markingSound.SetActive(false);
81.                 }
82.             }
83.             else
84.             {
85.                 laserObject.SetActive(false);
86.                 if (isDrawing)
87.                 {
88.                     StopDrawing();
89.                     markingSound.SetActive(false);
90.                 }
91.             }
92.
93.             if (currentLineRenderer != null)
94.             {
95.                 currentLineRenderer.transform.position =
   parentObject.transform.position;
96.             }
97.         }
98.     }
99.
100.         public void SetLineWidthFromSlider(float value)
101.         {
```

```
102.             lineWidth = value;
103.             SetLineWidth(lineWidth);
104.         }
105.
106.         void HandleLaser()
107.         {
108.             Vector3 rayStart = laserOrigin.position;
109.             Vector3 rayDirection = laserOrigin.forward;
110.             laserLine.SetPosition(0, rayStart);
111.             RaycastHit hit;
112.
113.             if (Physics.Raycast(laserOrigin.position,
    laserOrigin.forward, out hit, Mathf.Infinity))
114.             {
115.                 laserLine.SetPosition(1, hit.point);
116.                 laserLine.material.color = hitColor;
117.             }
118.             else
119.             {
120.                 laserLine.SetPosition(1, rayStart +
    (rayDirection * 100f));
121.                 laserLine.material.color =
    defaultLaserColor;
122.             }
123.         }
124.
125.         void StartDrawing()
126.         {
127.             isDrawing = true;
128.             if (currentLineRenderer != null)
129.             {
130.                 MeshCollider collider =
    currentLineRenderer.gameObject.GetComponent<MeshCollider>();
131.                 if (collider != null)
132.                 {
133.                     collider.enabled = false;
```

```
134.                 }
135.             }
136.             currentLineRenderer = CreateNewLineRenderer();
137.             currentLineRenderer.material = lineMaterial;
138.             SetLineWidth(lineWidth);
139.             UpdateLine();
140.         }
141.
142.         void UpdateLine()
143.         {
144.             if (currentLineRenderer != null)
145.             {
146.                 RaycastHit hit;
147.                 if (Physics.Raycast(laserOrigin.position,
    laserOrigin.forward, out hit, Mathf.Infinity))
148.                 {
149.                     currentLineRenderer.positionCount++;
150.                     int index =
    currentLineRenderer.positionCount - 1;
151.                     currentLineRenderer.SetPosition(index,
    hit.point);
152.                 }
153.             }
154.         }
155.
156.         void StopDrawing()
157.         {
158.             isDrawing = false;
159.             if (currentLineRenderer != null)
160.             {
161.                 createdMeshObjects.Add(GenerateMeshCollider(
    currentLineRenderer));
162.             }
163.             currentLineRenderer = null;
164.         }
165.
```

```
166.         GameObject GenerateMeshCollider(LineRenderer
   lineRenderer)
167.         {
168.             GameObject meshObject = new
   GameObject("MeshObject");
169.             meshObject.transform.SetParent(parentObject.tran
   sform);
170.             meshObject.layer =
   LayerMask.NameToLayer("Marking");
171.
172.             MeshFilter meshFilter =
   meshObject.AddComponent<MeshFilter>();
173.             MeshRenderer meshRenderer =
   meshObject.AddComponent<MeshRenderer>();
174.
175.             Mesh mesh = new Mesh();
176.             lineRenderer.BakeMesh(mesh, true);
177.             meshFilter.sharedMesh = mesh;
178.
179.             meshRenderer.sharedMaterial =
   lineRenderer.material;
180.
181.             Destroy(lineRenderer.gameObject);
182.
183.             return meshObject;
184.         }
185.
186.         void TrimUndoList()
187.         {
188.             while (undoLines.Count > maxUndoSteps)
189.             {
190.                 undoLines.RemoveAt(undoLines.Count - 1);
191.             }
192.         }
193.
194.         public void SetLineWidth(float width)
```

```
195.         {
196.             if (currentLineRenderer != null)
197.             {
198.                 currentLineRenderer.widthMultiplier =
    Mathf.Max(width, 0.0f);
199.             }
200.         }
201.
202.         public void Undo()
203.         {
204.             if (createdMeshObjects.Count > 0)
205.             {
206.                 int lastIndex = createdMeshObjects.Count -
    1;
207.                 Destroy(createdMeshObjects[lastIndex]);
208.
209.                 createdMeshObjects.RemoveAt(lastIndex);
210.             }
211.         }
212.
213.         public void ResetAllLines()
214.         {
215.             foreach (var meshObject in createdMeshObjects)
216.             {
217.                 Destroy(meshObject);
218.             }
219.             createdMeshObjects.Clear();
220.         }
221.
222.         private IEnumerator VibrateController(float
    duration, float intensity)
223.         {
224.             OVRInput.SetControllerVibration(intensity,
    intensity, OVRInput.Controller.RTouch);
225.             yield return new WaitForSeconds(duration);
```

```
226.             OVRInput.SetControllerVibration(0, 0,
   OVRInput.Controller.RTouch);
227.         }
228.
229.         LineRenderer CreateNewLineRenderer()
230.         {
231.             GameObject newLineObject = new
   GameObject("LineRenderer");
232.             newLineObject.transform.SetParent(parentObject.t
   ransform);
233.             LineRenderer lineRenderer =
   newLineObject.AddComponent<LineRenderer>();
234.             lineRenderer.material = lineMaterial;
235.             lineRenderer.positionCount = 0;
236.             return lineRenderer;
237.         }
238.     }
```

Rentang Kode	**Fungsi/Segmen Kode**	**Deskripsi**
1-5	Deklarasi *library*	Direktif menggunakan untuk pustaka yang diperlukan
6-7	Deklarasi Kelas	Deklarasi kelas *SnapModeMarking*
9-18	Deklarasi Variabel	Deklarasi variabel publik untuk objek *marker*, ketebalan, dan laser
20-25	Deklarasi Variabel Privat	Deklarasi variabel privat untuk pengelolaan garis dan status menggambar
27-31	Deklarasi Variabel laser	Deklarasi atribut untuk laser
33-36	Deklarasi Variabel *feedback* kontroler	Deklarasi atribut untuk *feedback* kontroler

38-51	*Start*()	Inisialisasi komponen dan *slider* lebar garis, serta pengaturan objek laser
53-98	*Update*()	Mengelola logika menggambar garis saat tombol ditekan dan *transformer* aktif, serta mengelola laser
100-104	*SetLineWidth FromSlider*(*float value*)	Mengatur lebar garis berdasarkan nilai *slider*
Rentang Kode	**Fungsi/Segmen Kode**	**Deskripsi**
106-123	*HandleLaser*()	Mengelola laser untuk menampilkan garis dan warna berdasarkan hitungan *raycast*
125-140	*StartDrawing*()	Memulai menggambar garis baru dan menonaktifkan *collider* jika ada
142-154	*UpdateLine*()	Memperbarui posisi garis saat menggambar berdasarkan hitungan *raycast*
156-164	*StopDrawing*()	Menghentikan menggambar garis dan menghasilkan *mesh collider* dari garis yang dibuat
166-184	*GenerateMesh Collider*(*LineRenderer lineRenderer*)	Menghasilkan objek *mesh collider* dari *line renderer* yang telah dibuat dan menghancurkan *line renderer* tersebut
186-193	*TrimUndoList*()	Memangkas daftar *undo* jika melebihi jumlah maksimum langkah *undo*
194-200	*SetLineWidth*(*float width*)	Mengatur lebar garis dari *line renderer* yang sedang aktif
202-211	*Undo*()	Menghapus objek *mesh* terakhir yang dibuat dari daftar dan menghapusnya dari tampilan

213-220	*ResetAllLines*()	Menghapus semua objek *mesh* yang dibuat dan mengosongkan daftar *undo*
222-227	*VibrateController* (*float duration, float intensity*)	Memberikan *feedback* kontroler saat menggambar garis
229-237	*CreateNewLine Renderer*()	Membuat dan menginisialisasi objek *line renderer* baru

Lampiran 4 Dokumentasi Pengambilan Data Testimoni