## LAPORAN PEMROGRAMAN VISUAL

## Dosen Pengampu :

Evianita Dwi Fajrianti, S.Tr.T., M.Tr.T., Ph.D

**Disusun oleh :**

1. Anaya Rizka Putri Novianti ( 5124521002 )
2. Muhammad Irsadul Ibad (5124521007)
3. Muhammad Afif Habiburrahman (5124521010)
4. Mayva Sofiana (5124521018)
5. Devi Hidayah (5124521022)

**TEKNOLOGI MULTIMEDIA BROADCASTING**

**DEPARTEMEN TEKNOLOGI MULTIMEDIA KREATIF**

**POLITEKNIK ELEKTRONIKA NEGERI SURABAYA**

**KAMPUS LAMONGAN**

**TAHUN 2025**

**LAPORAN UNITY PATHWAY**

**JUNIOR PROGRAMMING AS SOON AS POSSIBLE**

**BAB I**

* 1. **PENDAHULUAN**

Dalam dunia pengembangan game, unity merupakan *game engine* yang diimplementasikan secara luas untuk produksi berbagai macam aplikasi permainan, seperti :permainan 2D, 3D, Virtual Reality (VR), maupun Augmented Reality (AR). Unity menyediakan lingkungan pengembangan yang terpadu (Integrated Development Environment atau IDE) yang mana nantinya memungkinkan pengembang untuk merancang visual, memprogram logika, dan mengatur interaksi objek secara real-time. Bahasa pemrograman yang sering digunakan dalam Unity adalah C# (C-Sharp), yang berperan sebagai tulang punggung dalam mengendalikan perilaku objek dan sistem di dalam game engine.

Dalam konteks Unity Pathway Junior Programmer, dasar teori yang digunakan meliputi variabel komponen dan GameObject, yang menjadi elemen dasar dalam Unity. Setiap objek di dalam scene Unity disebut GameObject, dan setiap GameObject dapat beberapa komponen yang menentukan fungsi dan sifatnya, seperti: Transform (untuk posisi dan rotasi), Rigidbody (untuk fisika), atau Collider (untuk deteksi tabrakan). Hubungan antarobjek diatur dalam hierarki yang menyerupai struktur pohon, memudahkan pengembang dalam mengelola elemen-elemen kompleks di dalam dunia virtual.

Selain itu, pemrograman dalam Unity juga berfokus pada sistem event dan input, yaitu tentang bagaimana permainan merespons tindakan pemain seperti menekan tombol, menggerakkan mouse, atau menyentuh layar. Respon tersebut nantinya diatur dalam metode Update() atau FixedUpdate() yang berjalan secara berulang (loop) mengikuti frame rate permainan. Konsep ini dikenal sebagai Game Loop, yaitu siklus utama yang mengatur keseluruhan aktivitas dalam game, mulai dari pembaruan logika, pergerakan objek, hingga tampilan visual yang nampak pada pada layar.

Dengan memahami konsep dasar teor ini, mahasiswa tidak hanya mempelajari langkah teknis dalam mengikuti Unity Pathway, tetapi juga memperoleh pemahaman konseptual tentang bagaimana sistem interaktif dirancang dan dijalankan. Pengetahuan ini menjadi fondasi penting dalam pengembangan proyek-proyek lanjutan di bidang *game development*, simulasi interaktif, atau aplikasi multimedia dengan teknologi real-time.

* 1. **TUJUAN**
* Memahami cara kerja unity.
* Mampu mengimplementasikan unity.
* Membangun kerangka berpikir logis dan *problem solving* dalam perancangan proyek interaktif sederhana.
* Mengenal konsep pemrograman berorientasi objek di unity.
  1. **PERALATAN**
* Laptop.
* Unity hub.
* Notepad ++.

**BAB II**

**2. 1 GAMBARAN UMUM UNITY PATHWAY JUNIOR PROGRAMER.**

**Pathway Junior Programming** dibagi menjadi beberapa unit / mission. Di halaman pathway, unit yang terlihat adalah :

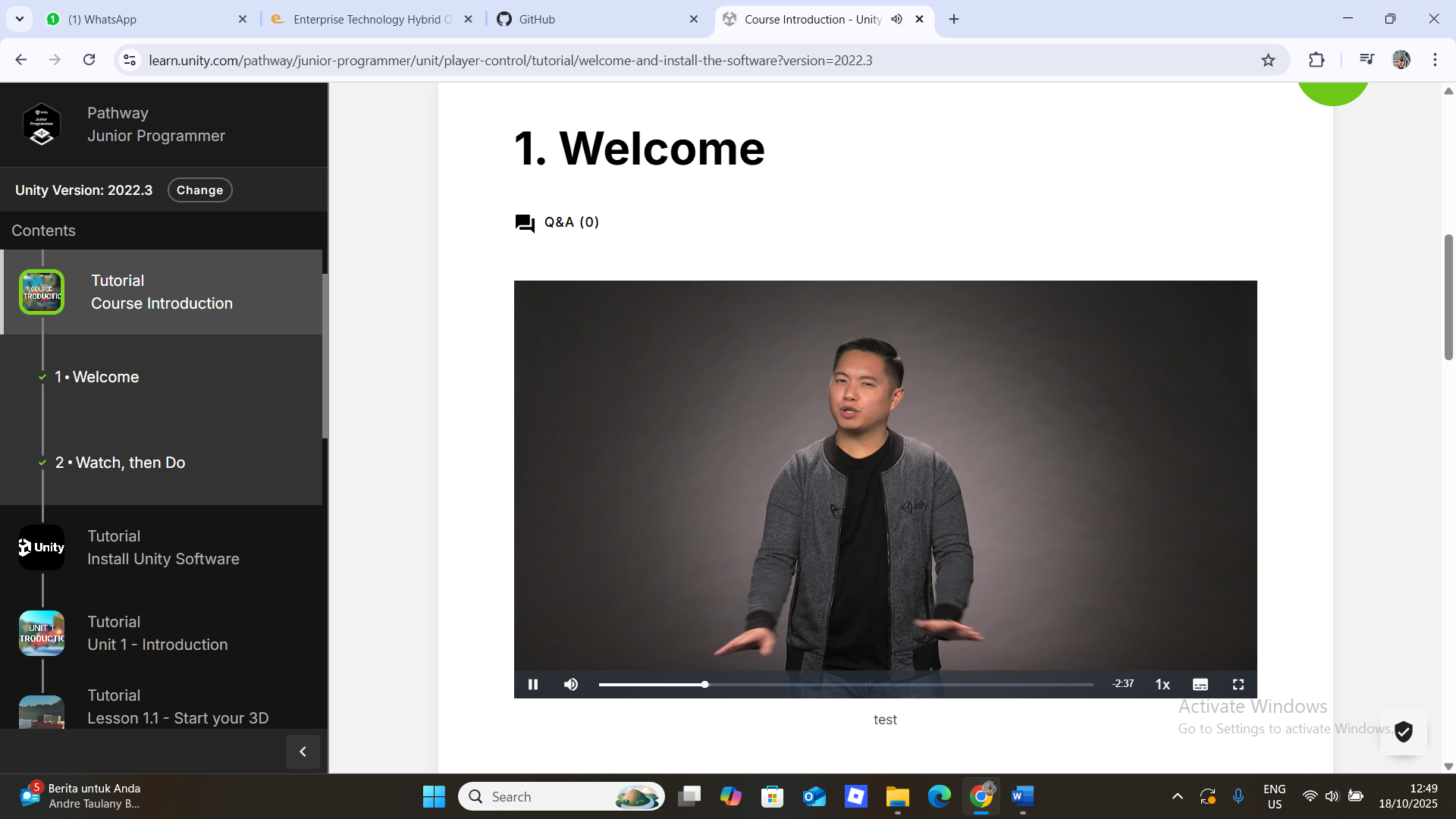
1. Player Control
2. Basic Gameplay
3. Sound and Effects
4. Gameplay Mechanics
5. User Interfce
6. Manage Scene Flow and Data
7. Apply Object-Oriented Principles

**Mission 1 : Player Control**

Misi pertama berfungsi sebagai landasan awal dalam pemrograman Unity. Dimana akan terdapat tutorial pembelajaran dasar-dasar bahasa C# dan bagaimana membuat script untuk berinteraksi dengan dunia 3D. Yang pada intinya, misi ini mengiplementasikan kontrol pemain yang responsif, kemudian juga belajar cara memproses input dari keyboard dan menerjemahkannya menjadi gerakan, rotasi, atau aksi pada objek dalam dunia 3D.

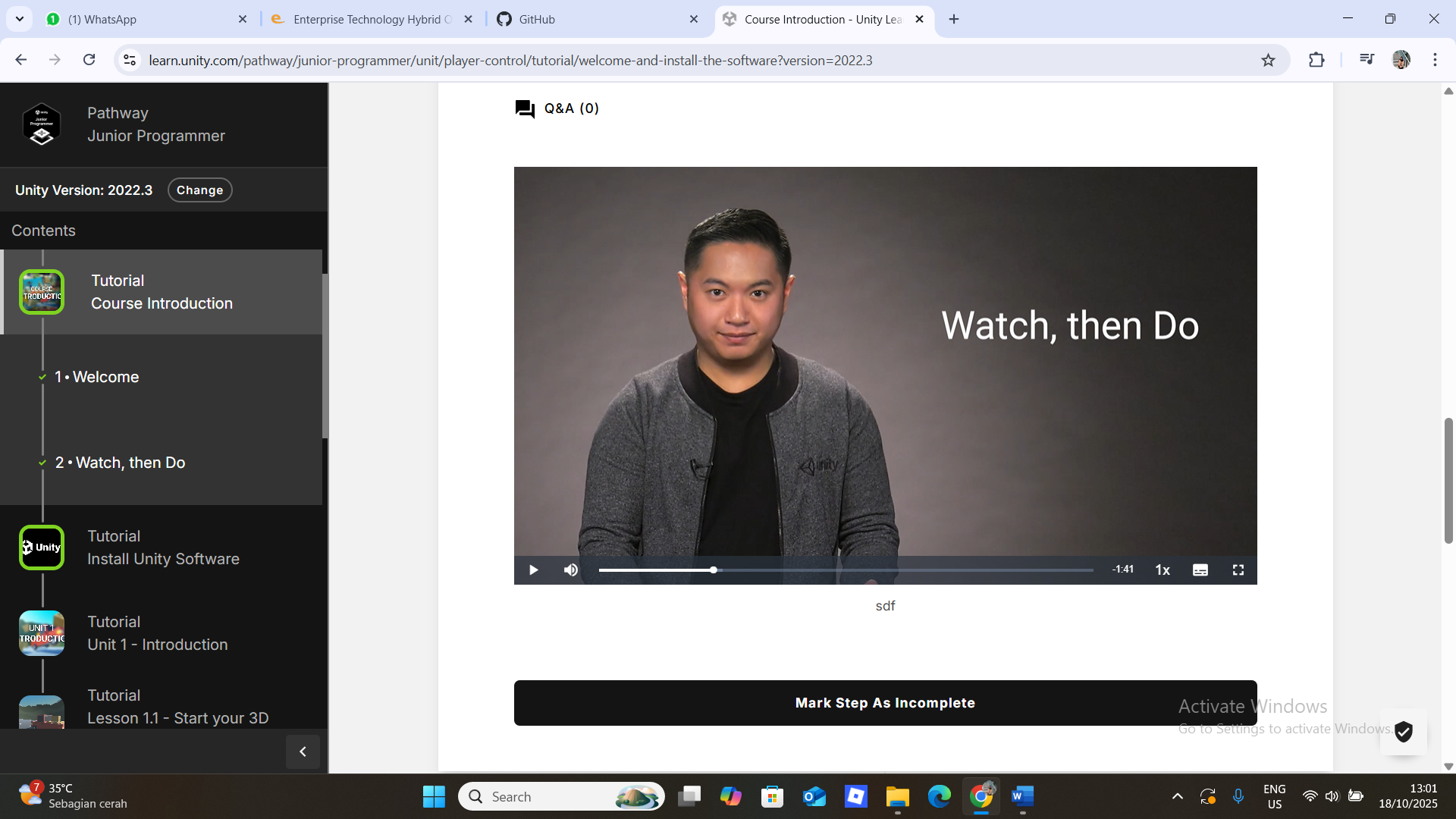
* **Tutorial Course Introduction**

1. **Welcome**



Pada **Tutorial Course Introduction** di bagian welcome**,** ini menjelaskan mengenai sebuah tutorial yang dirancang untuk mengajarkan pemrograman secara praktis menggunakan Unity dan C#, dua keterampilan penting di industri teknologi, game, dan Augmented Reality (AR). Melalui pembuatan lima proyek prototipe interaktif yang akan melatih dalam menulis kode dari awal, memahami fitur-fitur utama Unity, serta menerapkan konsep pemrograman dalam pengembangan aplikasi lintas platform seperti Android, iOS, PC, konsol, VR, dan AR.

1. **Watch, Then do**

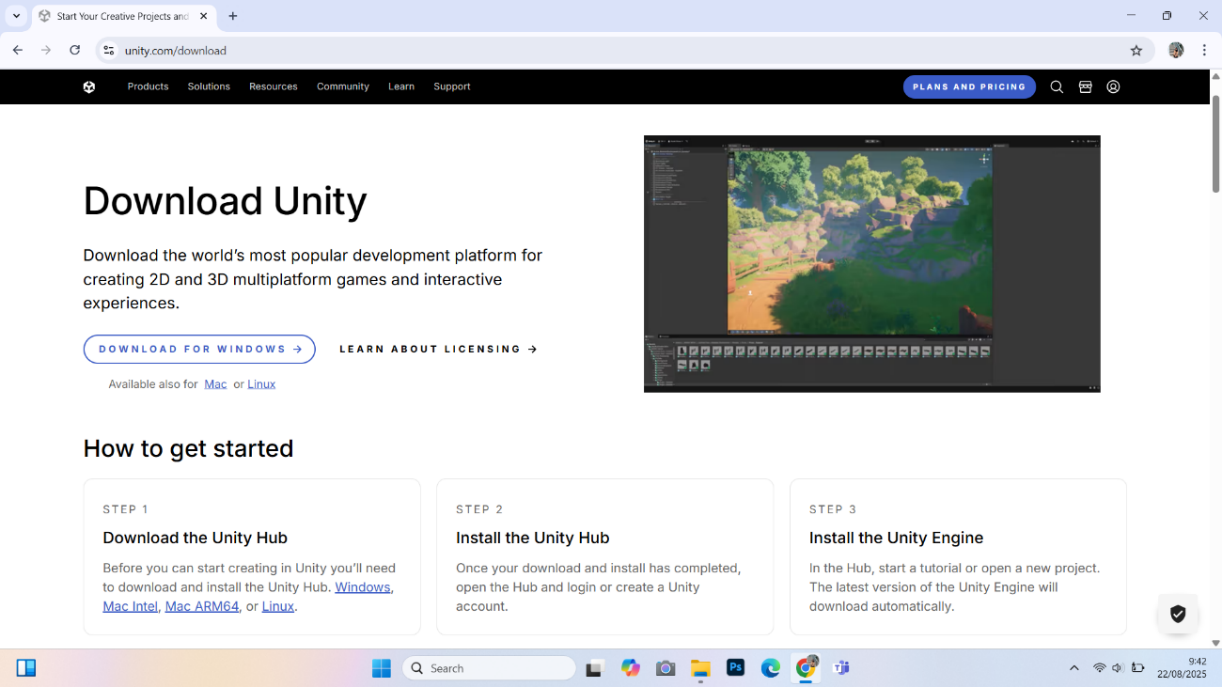


Bagian **Watch, Then Do** memberikan kesempatan untuk belajar mandiri dengan meniru langkah dari video, namun metode ini sering menyulitkan karena harus berpindah antara tampilan video dan proyek, sehingga berisiko melewatkan bagian penting. Cara yang lebih efektif adalah menonton seluruh video terlebih dahulu untuk memahami alur dan tujuannya, lalu mempraktikkannya secara bertahap. Dengan cara ini, proses belajar menjadi lebih terarah, pemahaman meningkat, dan hasil yang diperoleh lebih optimal.

* **Tutorial Install Unity Software**

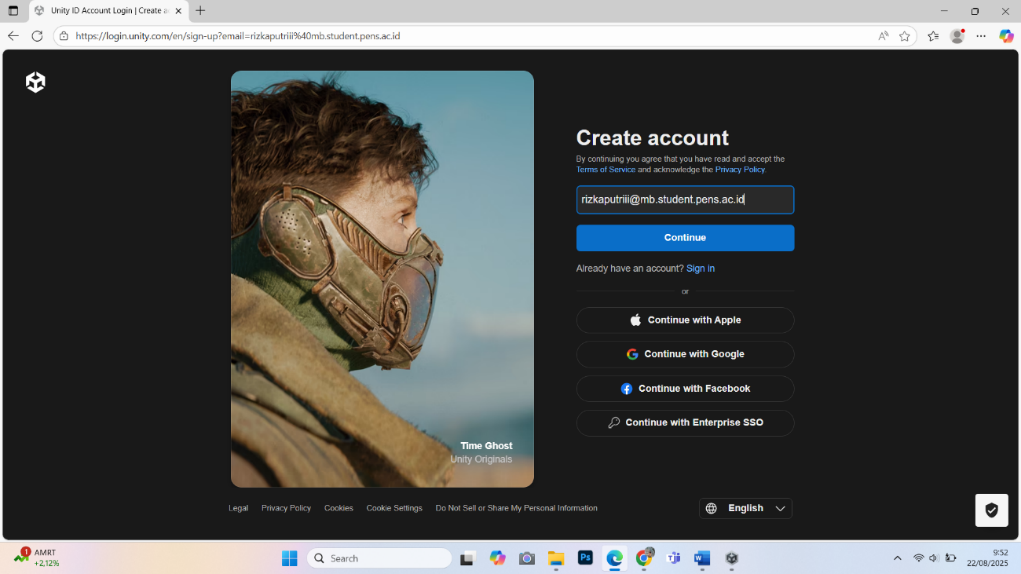
**Langkah 1 :**

Buka google, kemudian ketik di pencarian “Unity Hub” . Buka website resmi Unity, kemudian klik “Download For Windows”.

****

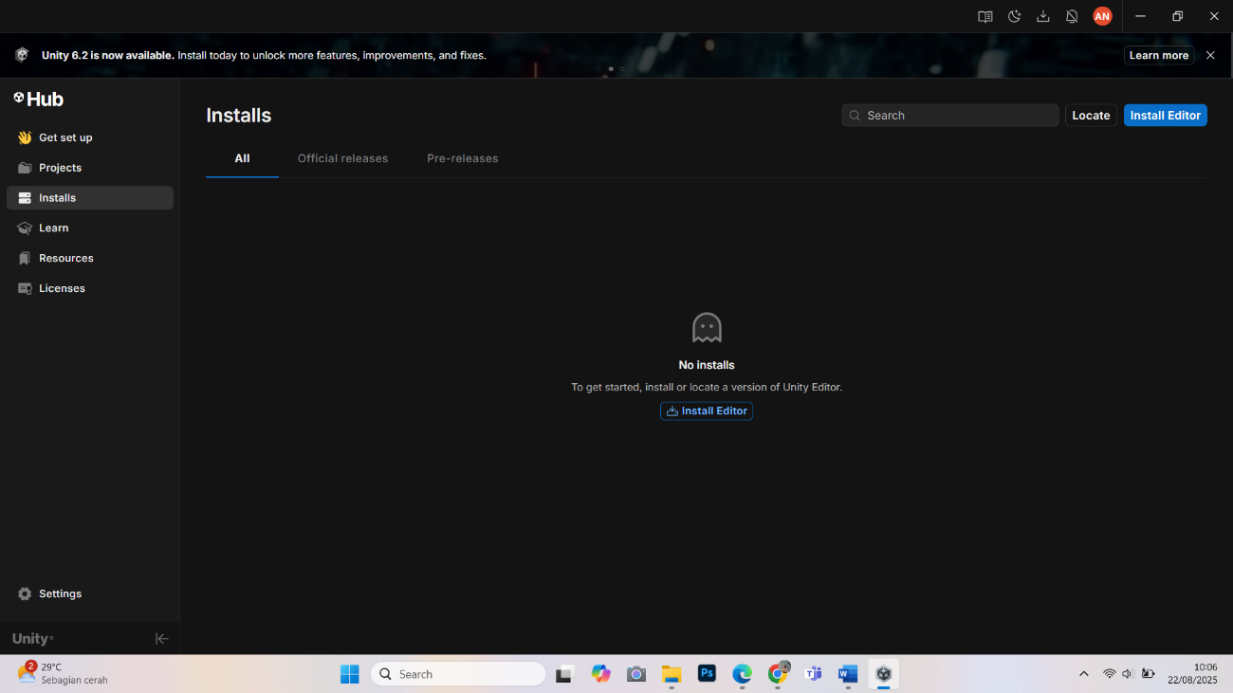
**Langkah 2 :**

Login Unity, pilih “Create Account” untuk membuat akun baru. Masukkan email yang akan digunakan untuk membuat akun Unity. Tambahkan user, Full name dan password. Password memiliki ketentuan maximal 8 karakter dengan campuran huruf besar, angka dan simbol. Lakukan verifikasi akun setelah login.

****

**Langkah 3 :**

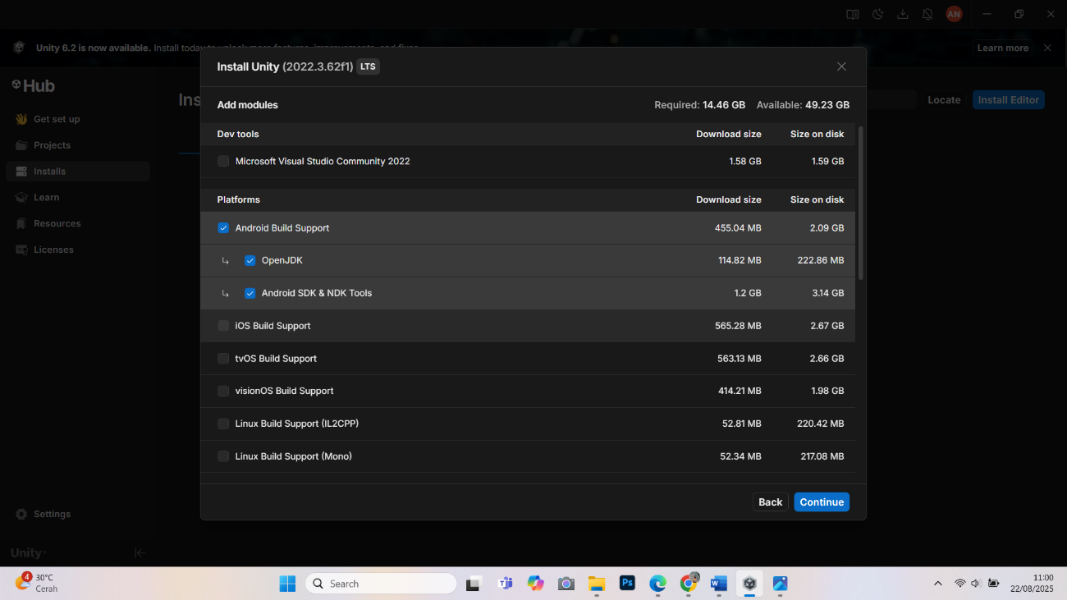
Masuk ke aplikasi Unity dan klik “Install Editor”

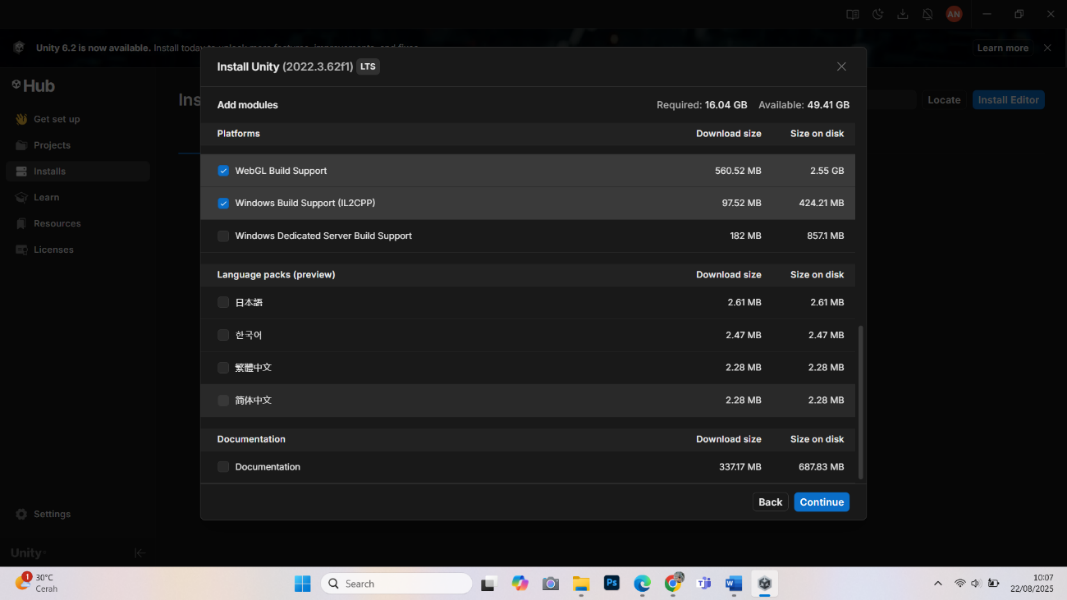
****

**Langkah 4 :**

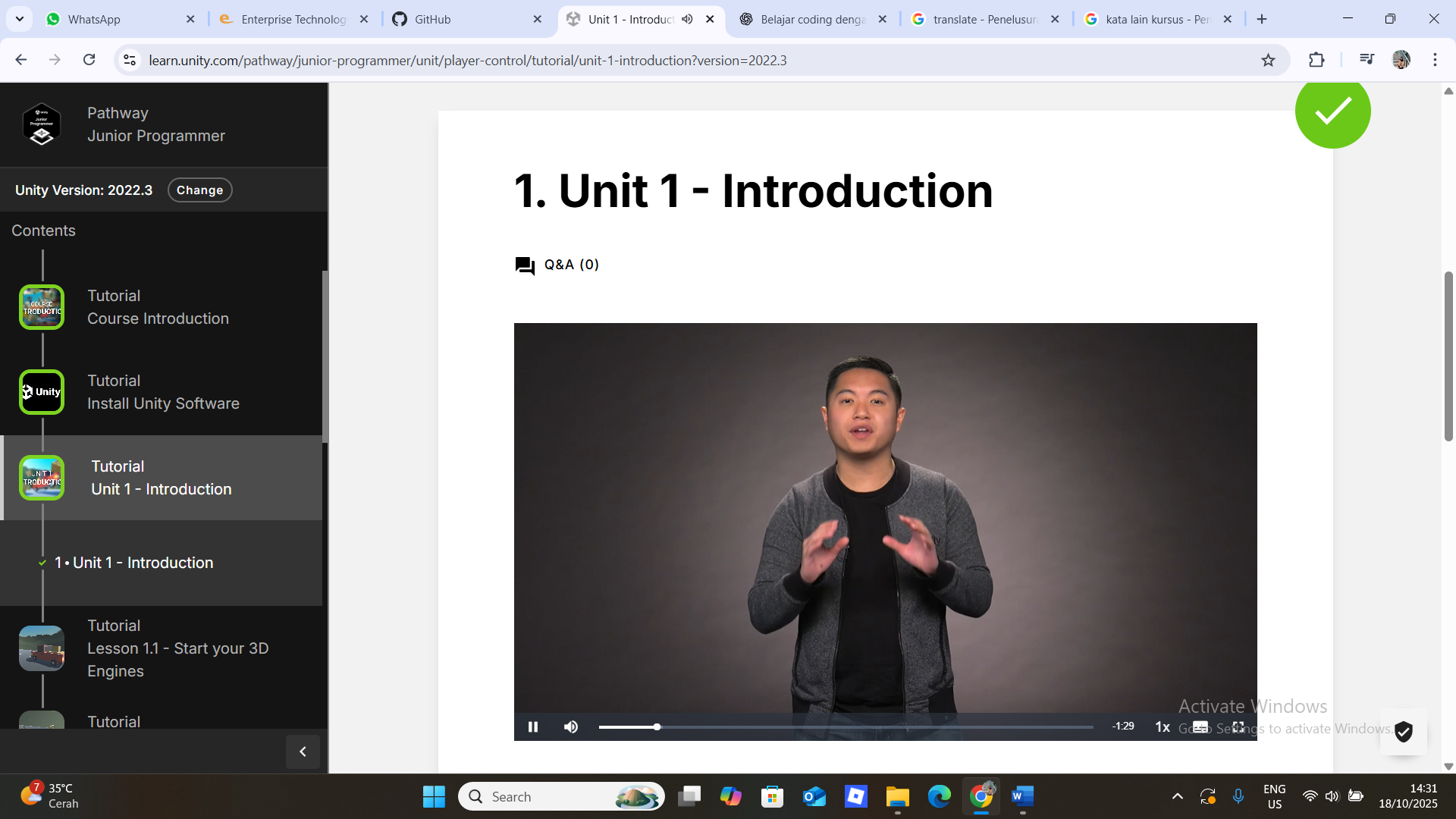
Pilih Unity versi 2022, apabila menggunakan user IOS centang IOS Build Support, apabila menggunakan user andorid, centang Android Build Support. Selain itu juga centang Webgl Build Support dan Windows Build Support (IL2CPP). Kemudian klik continue untuk mulai menginstal. Setelah semuanya sudah terinstal Unity sudah bisa digunakan

****

****

****

* **Tutorial Unit 1 – Introduction**



Pada tutorial Unit 1 ini dibuat sebuah prototipe simulator mengemudi, di mana pemain mengendalikan kendaraan untuk menghindari rintangan di jalan. Melalui pembuatan prototipe ini dipelajari dasar-dasar pemrograman kontrol pemain, yaitu bagaimana sistem merespons masukan seperti penekanan tombol, gerakan mouse, atau penggunaan joystick, lalu menerjemahkannya menjadi aksi di lingkungan 3D. Kontrol pemain merupakan elemen penting dalam setiap game karena memungkinkan terjadinya interaksi yang bermakna. Pemahaman terhadap konsep ini menjadi dasar dalam menciptakan berbagai bentuk kontrol yang menarik pada proyek interaktif berikutnya.

* **Tutorial Lesson 1.1 – Start your 3D Engines**

1. **Make a courses folder and new project**

Tahap awal dimulai dengan membuka Unity Hub dan membuat proyek baru menggunakan template 3D (Core), karena proyek ini memanfaatkan elemen tiga dimensi seperti objek dan lingkungan ruang. Beri nama proyek sesuai kebutuhan, tentukan lokasi penyimpanan, lalu tekan Create Project. Setelah editor terbuka, simpan scene pertama ke folder Assets/Scenes dengan menekan Ctrl + S agar setiap perubahan tersimpan dengan aman.

1. **Import assets and open prototype 1**

Unduh dan mengekstrak Prototype 1 Starter Files agar semua aset siap digunakan. File hasil ekstraksi kemudian dimasukkan ke dalam proyek agar seluruh komponen siap digunakan. Setelah impor selesai, scene utama yang digunakan adalah Prototype 1, sementara scene bawaan seperti Sample Scene dihapus agar tampilan proyek menjadi lebih rapi. Pada tahap ini, dilakukan pengecekan dengan melihat area sekitar di Scene View untuk memastikan seluruh aset telah terpasang dengan baik dan proyek siap dikembangkan lebih lanjut.

1. **Add your vehicle to the scene**

Langkah ini dimulai dengan membuka Project Window, lalu masuk ke folder Assets, pilih Course Library, dan buka bagian Vehicles. Pilih salah satu kendaraan dan seret ke Hierarchy agar muncul di Scene. Gunakan klik kanan dan tombol WASD untuk mengatur pandangan, lalu tekan F agar kamera terfokus pada kendaraan. Gunakan scroll wheel untuk zoom, tahan scroll wheel untuk menggeser pandangan, dan tekan Alt + klik kiri untuk memutar kamera. Jika terjadi kesalahan, tekan Ctrl + Z untuk membatalkan perubahan.

1. **Add an obstacle and reposition it**

Langkah berikutnya dilakukan dengan membuka Course Library, kemudian memilih Obstacles dan menyeret salah satu rintangan ke dalam tampilan Scene. Setelah rintangan muncul, buka Inspector dan pada bagian Transform, pilih opsi Reset untuk mengembalikan posisinya ke titik awal. Selanjutnya, atur nilai posisi menjadi x = 0, y = 0, dan z = 25 agar rintangan berada di tempat yang diinginkan. Terakhir, buka jendela Hierarchy dan ubah nama kedua objek menjadi Vehicle dan Obstacle agar mudah dikenali selama proses pengembangan proyek.

1. **Locate your camera and run the game**

Pada tahap ini, kamera difokuskan terlebih dahulu agar tampilan di Scene lebih terarah dan mudah diamati. Setelah posisi kamera sesuai, permainan dijalankan melalui tombol Play di bagian atas Unity untuk melihat sudut pandang yang akan digunakan dalam game. Melalui cara ini, pengguna dapat memahami bagaimana tampilan permainan terlihat langsung dari perspektif kamera di dalam editor.

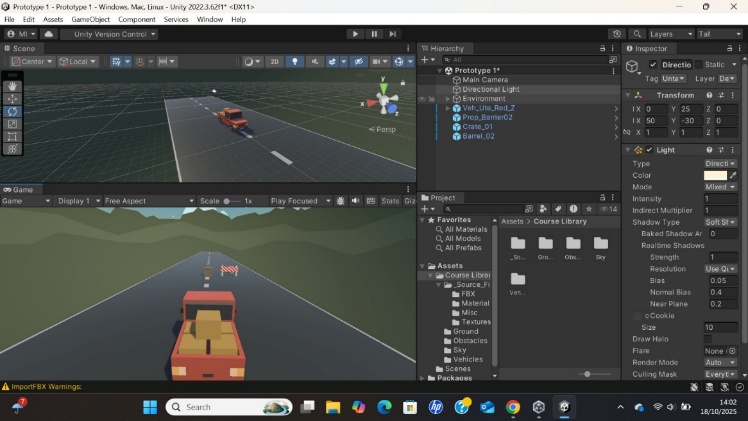
1. **Move the camera behind the vehicle**

Kamera kemudian diposisikan di belakang kendaraan dengan sudut pandang sedikit menurun agar seluruh bagian kendaraan terlihat dengan jelas dari atas. Penyesuaian dilakukan secara presisi untuk mendapatkan tampilan yang ideal, memastikan sudut pandang kamera memberikan perspektif terbaik terhadap pergerakan kendaraan di dalam scene.

1. **Customize the interface layout**

Tampilan kerja kemudian diatur ulang agar lebih nyaman digunakan. Area kerja dibuat lebih tinggi dengan memilih opsi Tall, sementara jendela Game View dipindahkan ke bawah Scene View supaya proses pengeditan dan hasil game dapat terlihat bersamaan. Di jendela Project, tampilan file disusun dalam format satu kolom agar lebih rapi. Setelah penyesuaian selesai, pengaturan tata letak ini disimpan dengan nama My Layout agar mudah digunakan kembali di kemudian hari.

1. **Lesson Recap**

****

* **Tutorial Lesson 1.2 – Pedal to the Metal**

1. **Create abd apply your first script**

Gambar tersebut memperlihatkan proses penambahan script pengendali pada proyek Unity. Sebuah folder Scripts dibuat untuk menyimpan file kode, kemudian ditambahkan script C# bernama PlayerController yang berfungsi mengatur pergerakan kendaraan. Script yang telah dibuat kemudian dipasangkan pada objek kendaraan dengan menyeretnya langsung ke objek tersebut di jendela Hierarchy, sehingga script muncul sebagai komponen di panel Inspector dan siap dijalankan untuk mengontrol perilaku kendaraan dalam permainan.

1. **Add a comment in the Update () method**

Proses awal pengisian skrip PlayerController dilakukan dengan menambahkan komentar “// Move the vehicle forward” pada bagian Update(). Komentar ini berfungsi sebagai penanda area kode yang nantinya digunakan untuk mengatur pergerakan kendaraan ke depan, sebelum logika utama dikembangkan lebih lanjut.

1. **Give the vehicle a forward motion**

Pada tahap ini, bagian Update() diisi dengan perintah transform. Translate(0, 0, 1); yang berfungsi untuk menggerakkan kendaraan maju secara terus-menerus di arah sumbu Z. Baris kode ini menjadi dasar pergerakan objek dalam game, di mana nilai yang dimasukkan menentukan arah dan kecepatan gerakan kendaraan.

1. **Use a Vectore 3 to move forward**

Kode pada bagian ini disempurnakan dengan mengganti nilai (0, 0, 1) menjadi Vector3.forward. Perubahan ini membuat instruksi lebih efisien dan mudah dibaca, karena Vector3.forward secara langsung merepresentasikan arah maju pada sumbu Z di Unity.

1. **Customize the vehicle’s speed**

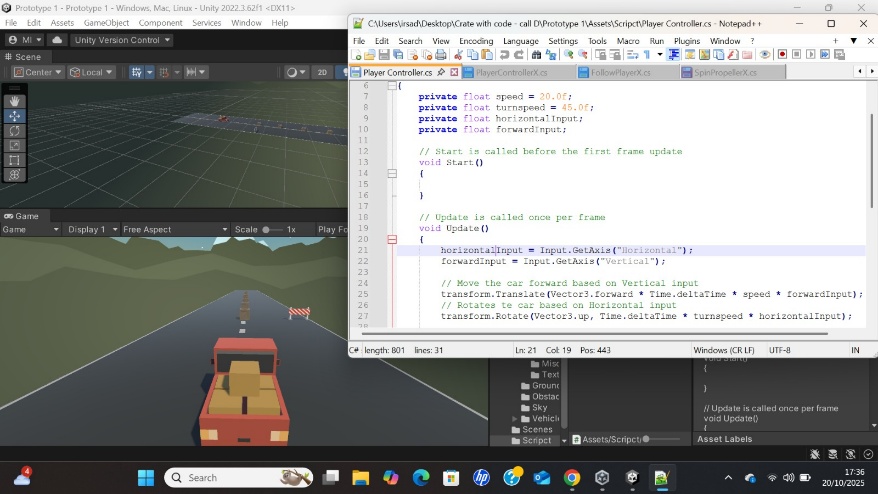
Bagian ini menambahkan Time.deltaTime dan angka kecepatan 20 ke dalam perintah Translate, sehingga gerakan kendaraan menjadi lebih halus dan konsisten di setiap frame. Penyesuaian ini memastikan kecepatan kendaraan tetap stabil meskipun performa game berbeda-beda.

1. **Add Rigidbody components to objets**

Pada bagian ini, objek Vehicle dan Obstacle diberi komponen Rigidbody melalui panel Inspector agar keduanya dapat berinteraksi menggunakan sistem fisika Unity. Komponen ini memungkinkan objek mengalami gaya, gravitasi, serta tumbukan yang lebih realistis. Setelah menambahkannya, nilai mass pada masing-masing objek diatur agar mendekati berat sebenarnya dalam satuan kilogram. Penyesuaian ini membuat simulasi pergerakan dan benturan di dalam permainan terlihat lebih alami dan seimbang.

1. **Duplicate and position the obstacles**

Langkah ini berfokus pada penataan dan penggandaan rintangan di dalam scene. Objek **Obstacle** dipindahkan ke bagian bawah daftar di Hierarchy agar lebih mudah dikelola. Setelah itu, objek tersebut digandakan beberapa kali menggunakan perintah **Ctrl + D**, lalu setiap salinan diposisikan sedikit menjauh pada sumbu **Z** untuk menciptakan deretan rintangan di jalur kendaraan. Proses ini diulang beberapa kali hingga terbentuk banyak rintangan

1. **Lesson Recap**

* **Tutorial Lesson 1.3 – High Speed Case**

1. **Add a speed variable for your vehicle**

Langkah pertama adalah menambahkan sebuah variabel baru bernama speed ke dalam skrip kendaraan. Variabel ini nantinya berfungsi untuk mengontrol seberapa cepat objek kendaraan bergerak di dalam game. Biasanya tipe datanya berupa float agar bisa menampung angka desimal. Nilai speed ini bisa diatur di Inspector Unity agar mudah diubah tanpa mengedit kode. Dengan cara ini, kamu dapat menguji berbagai tingkat kecepatan dan menemukan nilai yang terasa paling realistis atau sesuai dengan kebutuhan permainan.

1. **Create a new script for the camera**

Tahap berikutnya adalah membuat skrip baru yang khusus menangani perilaku kamera. Skrip ini bertujuan agar kamera dapat mengikuti kendaraan dengan cara yang dinamis dan halus. Dengan memisahkan logika kamera dari logika kendaraan, proyekmu menjadi lebih terstruktur dan mudah dikelola. Skrip kamera ini akan memudahkan kamu untuk menambahkan efek sinematik atau sistem follow camera yang realistis di kemudian hari.

1. **Add an offset to the camera position**

Agar kamera tidak menempel langsung pada kendaraan, kamu perlu menambahkan offset pada posisinya. *Offset* adalah jarak atau pergeseran antara posisi kendaraan dan posisi kamera. Biasanya offset diletakkan di belakang dan sedikit di atas kendaraan, supaya pemain dapat melihat seluruh kendaraan serta area di depannya dengan jelas. Dengan menambahkan *offset*, tampilan permainan menjadi lebih nyaman dan profesional.

1. **Make the offse into a Vector 3 variable**

Untuk mengatur offset secara fleksibel di ruang tiga dimensi, offset tersebut dibuat menjadi variabel dengan tipe data **Vector3**. **Vector3** memungkinkan untuk mengatur nilai pada sumbu X, Y, dan Z . Dengan menjadikannya variabel publik, kamu bisa menyesuaikan posisi kamera secara visual melalui Inspector, tanpa harus membuka skrip. Ini juga membantu saat kamu ingin mencoba berbagai sudut pandang kamera.

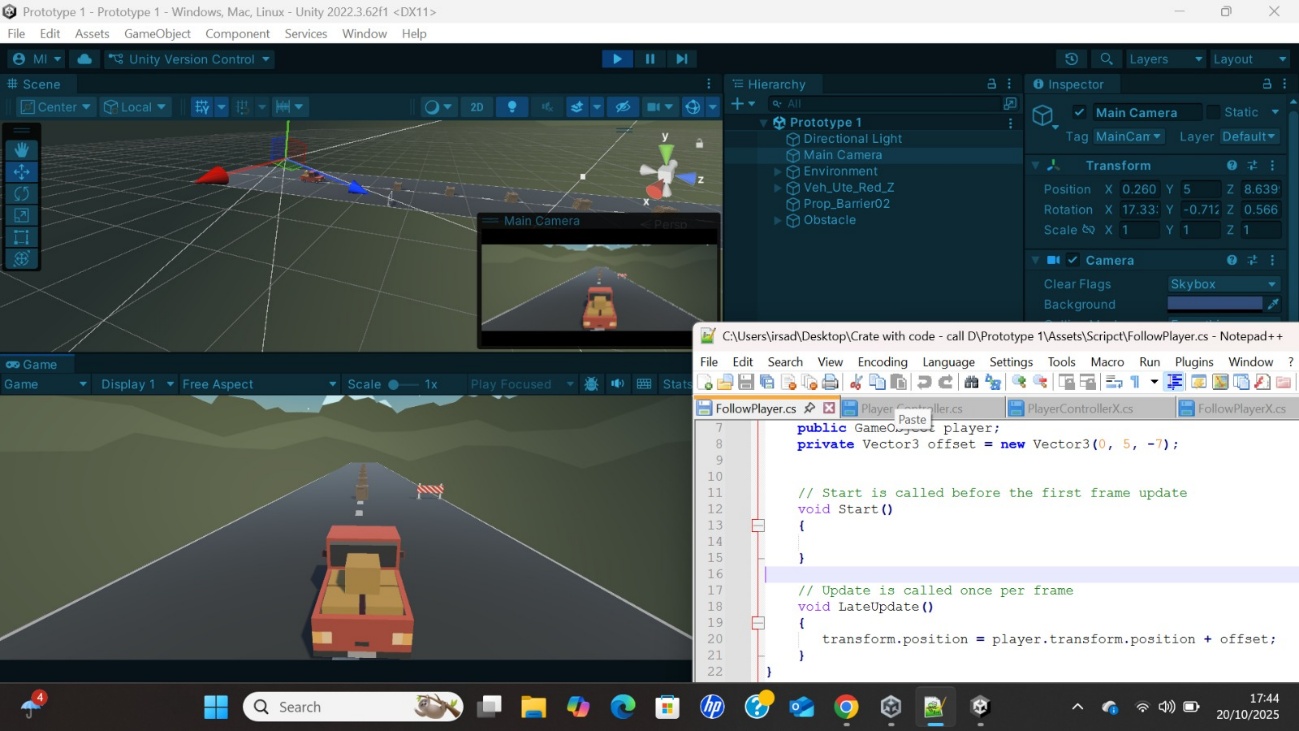
1. **Smooth the camera with LateUpdate**

Pergerakan kamera perlu dibuat halus agar tidak terjadi gerakan tiba-tiba atau patah-patah. Di sinilah fungsi **LateUpdate()** digunakan. ***LateUpdate*** dijalankan setelah semua perhitungan Update selesai, sehingga kamera dapat menyesuaikan posisinya setelah kendaraan bergerak. Dengan teknik ini, kamera akan tampak mengikuti kendaraan secara lembut, menciptakan kesan profesional dan meningkatkan pengalaman visual pemain.

1. [**Edit the playmode tint color**](https://learn.unity.com/pathway/junior-programmer/unit/player-control/tutorial/1-3-make-the-camera-follow-the-vehicle-with-variables?version=2022.3#658f3f17edbc2a2581f45605)

Langkah ini lebih ke aspek kenyamanan pengembangan. Unity memungkinkan kamu mengubah warna *Play Mode Tint* di Preferences, agar kamu tahu kapan sedang berada di mode Play. Ini penting karena sering kali developer lupa bahwa mereka sedang menjalankan game, lalu melakukan perubahan pada objek yang tidak akan tersimpan setelah *Play Mode* berhenti.

1. **Lesson Recap**



* **Tutorial Lesson 1.4 – Step into the Driver’s Seat**

1. **Allow the vehicle to move left/right**

Langkah pertama adalah membuat kendaraan bisa bergerak secara horizontal ke kiri dan ke kanan di samping gerakan maju yang sudah ada. Tujuannya adalah memberi pemain kendali penuh atas arah kendaraan, seperti saat mengemudikan mobil sungguhan. Dalam implementasinya, kamu akan menambahkan logika baru pada skrip kendaraan agar posisi kendaraan bisa berubah di sumbu X, sementara tetap mempertahankan kecepatan maju di sumbu Z. Ini menjadi dasar penting untuk sistem kontrol pemain (player control) yang lebih realistis.

1. **Base left/right movement on input**

Setelah kendaraan dapat bergerak ke arah horizontal, langkah berikutnya adalah menghubungkan gerakan tersebut dengan input dari pemain. Biasanya menggunakan fungsi *Input.GetAxis*(“Horizontal”) di Unity, yang mendeteksi tombol panah kiri/kanan atau A/D pada keyboard. Dengan begitu, kendaraan tidak hanya bergerak secara otomatis, tapi benar-benar merespons perintah pemain. Inilah yang menjadikan pengalaman bermain terasa interaktif dan intuitif pemain merasa seolah benar-benar memegang kemudi.

1. **Take control of the vehicle speed**

Tahap ini berfokus pada pengaturan kecepatan kendaraan agar bisa dikontrol secara dinamis, baik untuk akselerasi maupun deselerasi. Variabel speed yang sudah dibuat di pelajaran sebelumnya kini digunakan untuk mengatur seberapa cepat kendaraan melaju, dan bisa disesuaikan berdasarkan situasi permainan. Misalnya, dapat menambahkan batas kecepatan maksimum atau membuat kendaraan melambat saat tidak diberi input. Ini menambah nuansa realistis dan menjaga keseimbangan gameplay.

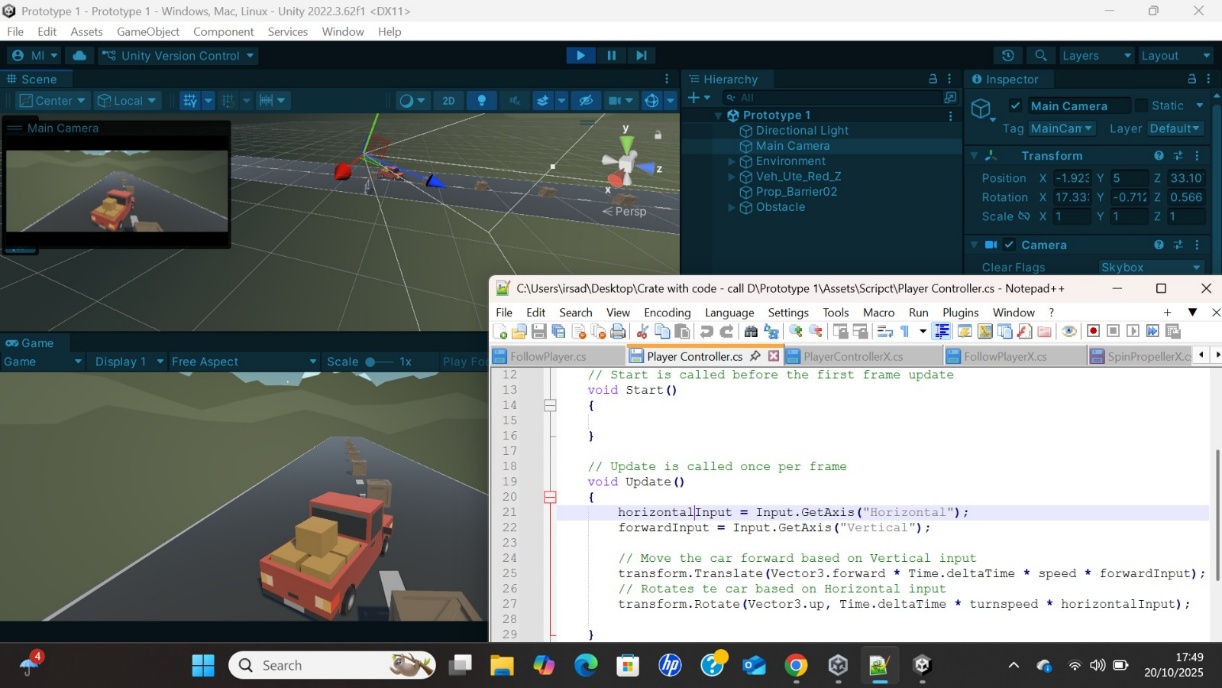
1. **Make vehicle rotate instead of slide**

Agar kendaraan terasa lebih alami saat berbelok, perlu dilakukan perubahan pada cara geraknya: bukan sekadar slide ke samping, tetapi rotate atau berputar mengikuti arah belokan. Teknik ini membuat mobil tampak benar-benar berbelok bagian depan mengarah ke arah tujuan, bukan hanya berpindah posisi. Biasanya, rotasi dilakukan dengan fungsi *transform.Rotate()* atau memodifikasi *Quaternion* agar pergerakan terlihat halus dan proporsional terhadap kecepatan kendaraan. Hasilnya, kontrol kendaraan terasa lebih realistis dan sinematik.

1. **Clean your code and hierarchy**

Setelah menambahkan banyak fitur baru, penting untuk membersihkan skrip dan hierarki pada proyek agar tetap rapi. Ini meliputi pemberian nama variabel yang konsisten, menghapus baris kode yang tidak dipakai, serta mengelompokkan objek di Hierarchy sesuai fungsinya (misalnya, memisahkan kamera, kendaraan, dan lingkungan). Langkah ini sangat fatal dalam pengembangan game, kerapian kode dan struktur proyek adalah kunci untuk efisiensi dan kolaborasi tim yang lancar.

1. **Lesson Recap**

****

**Tutorial Challenge 1 – Plane Programming**

1. **Challenge 1 Overview**

Pada tantangan pertama ini, peserta diminta untuk melakukan perbaikan pada berbagai masalah yang terjadi di sebuah pesawat dalam proyek Unity. Fokus inti tantangan adalah memahami logika pemrograman pada objek pesawat, termasuk arah gerak, kecepatan, rotasi, dan perilaku kamera yang mengikuti. Tujuannya agar peserta mampu mengasah dan mengimplementasikan kemampuan debugging, dan berpikir logis.

1. **Warning**

Sebelum memulai tantangan, peserta diingatkan untuk berhati-hati dalam melakukan perubahan pada kode. Beberapa kesalahan umum, seperti menghapus variabel penting atau menimpa nilai yang salah, dapat menyebabkan pesawat tidak berfungsi dengan benar. Peserta juga disarankan untuk selalu menyimpan salinan (backup) proyek sebelum memulai perbaikan, agar dapat memulihkan versi sebelumnya jika terjadi kesalahan fatal.

1. **The plane is going backward**

Masalah pertama yang harus diselesaikan adalah pesawat yang bergerak ke arah yang salah, yaitu mundur alih-alih maju. Hal ini biasanya disebabkan oleh nilai kecepatan yang negatif atau arah sumbu yang terbalik pada transformasi posisi. Peserta perlu memeriksa logika perhitungan transform.Translate() atau rigidbody.AddForce() dan memastikan arah gerak pesawat menggunakan sumbu positif Z agar pesawat melaju ke depan dengan benar.

1. **The plane is going to fast**

Masalah kedua berkaitan dengan kecepatan pesawat yang terlalu tinggi. Hal ini umumnya terjadi karena nilai variabel speed terlalu besar atau ada penambahan gaya (force) yang berlebihan dalam satu frame. Peserta perlu menyesuaikan nilai speed agar lebih realistis, serta membatasi kecepatan maksimum menggunakan fungsi seperti Mathf.Clamp(). Tujuannya agar pesawat tetap mudah dikendalikan dan tidak melewati batas fisika permainan.

1. **The plane is tilting automatically**

Masalah ketiga adalah pesawat yang miring secara otomatis tanpa input dari pemain. Ini biasanya disebabkan oleh rotasi awal objek yang tidak netral atau adanya skrip rotasi otomatis yang belum dihapus. Peserta perlu memastikan bahwa nilai rotasi hanya berubah ketika ada input pemain. Dengan demikian, pesawat hanya akan miring saat benar-benar dikendalikan, bukan secara otomatis.

1. **The camera is in front of the plane**

Masalah keempat adalah posisi kamera yang salah, yaitu berada di depan pesawat. Untuk memperbaikinya, peserta perlu menyesuaikan posisi kamera dengan menambahkan offset di belakang dan sedikit di atas pesawat. Dengan menggunakan variabel Vector3 offset, kamera dapat ditempatkan pada sudut pandang yang tepat, sehingga pemain dapat melihat seluruh bagian pesawat dengan jelas selama permainan berlangsung.

1. **The camera is not following the plane**

Masalah berikutnya adalah kamera yang tidak mengikuti gerakan pesawat. Hal ini biasanya terjadi karena skrip kamera belum mengacu pada posisi pesawat di setiap frame. Peserta perlu menambahkan logika di fungsi LateUpdate() agar posisi kamera selalu diperbarui mengikuti posisi pesawat ditambah dengan offset. Dengan cara ini, kamera akan bergerak secara halus dan seolah melekat di belakang pesawat.

1. **Bonus : The planes’s propeller does not spin**

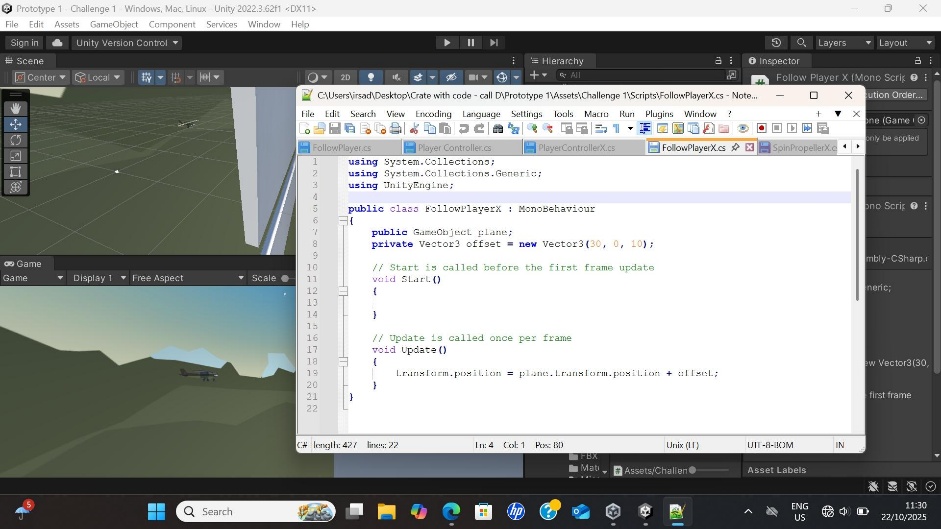
Sebagai tantangan tambahan, peserta diminta memperbaiki baling-baling pesawat (propeller) yang tidak berputar. Untuk mengatasinya, perlu ditambahkan logika rotasi pada objek propeller menggunakan transform.Rotate() dengan kecepatan konstan pada sumbu tertentu, biasanya sumbu Z atau Y. Langkah ini bersifat opsional, namun memberikan efek visual yang lebih realistis dan menarik pada pesawat.

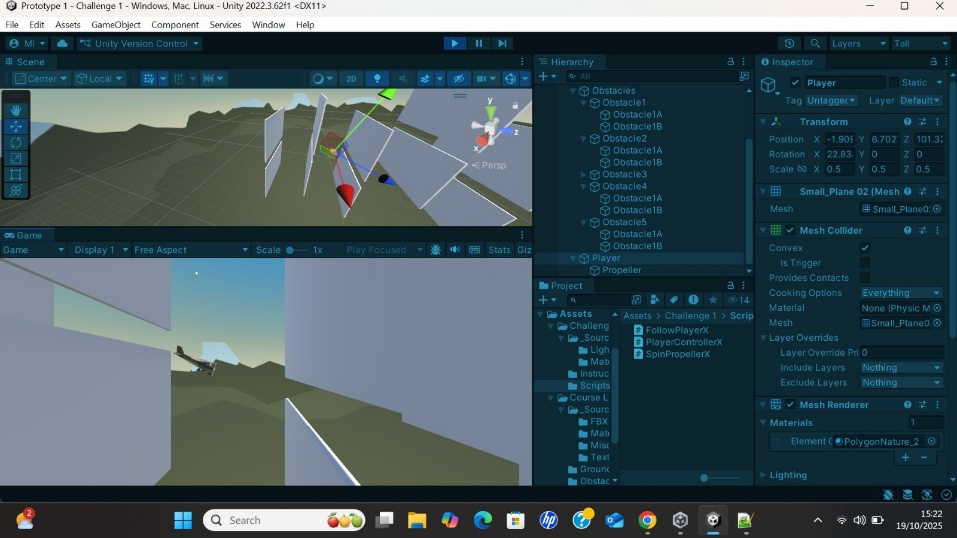
1. **(Optional) Hints**

Bagian ini menyediakan Petunjuk opsional bagi mereka yang kesulitan.Petunjuk tersebut membantu memahami penyebab utama dari setiap masalah, seperti bagaimana sistem transformasi bekerja di Unity, atau bagaimana cara mengatur kamera agar selalu mengikuti objek target. Petunjuk ini dapat digunakan sebagai panduan tanpa langsung memberikan jawaban akhir, agar peserta tetap belajar melalui eksplorasi.

1. **(Optional) Want even more challenges?**

Bagian terakhir memberikan opsi tambahan bagi peserta yang berminat untuk meningkatkan kemampuan mereka lebih jauh. Peserta didorong untuk menambahkan fitur tambahan, seperti efek suara mesin, sistem skor berdasarkan jarak terbang, atau efek partikel saat pesawat bergerak cepat. Tantangan ini bersifat opsional, namun bertujuan melatih kreativitas dan memperdalam pemahaman logika pemrograman dalam Unity.



****