BAB 3
PELAKSANAAN KERJA MAGANG

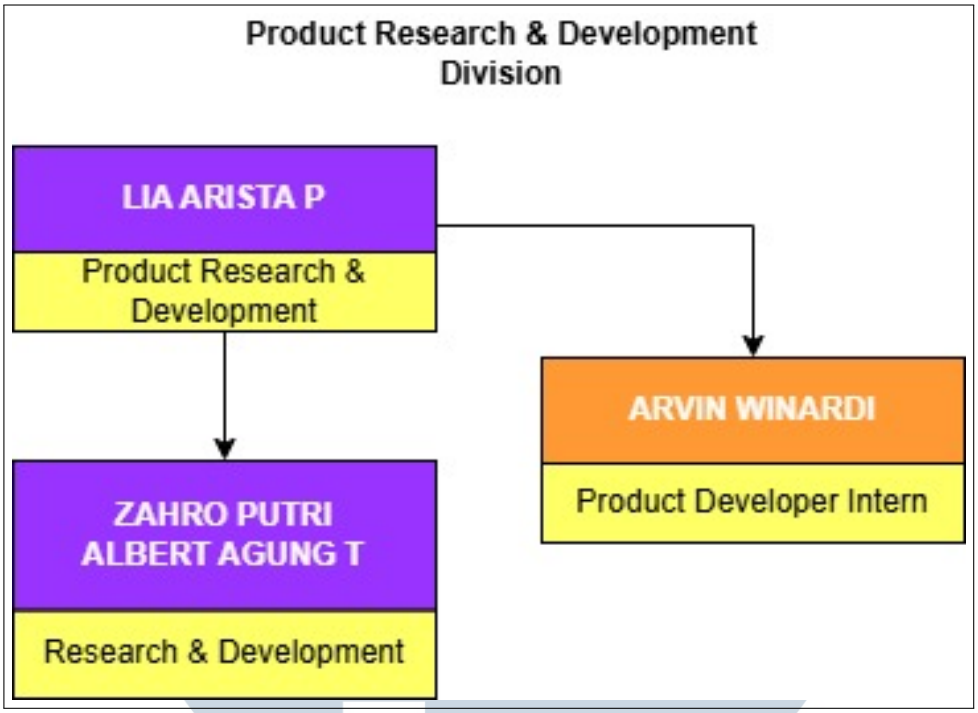
3.1 Kedudukan dan Organisasi

Dalam kerja magang di PT DigiKidz Indonesia, penulis memiliki peran dan tanggung jawab dalam koordinasi dan pelaksanaan berbagai tugas. Berikut ini adalah penjabarannya.

3.1.1 Kedudukan

Penulis menjalani kerja magang di PT DigiKidz Indonesia sebagai *Product Developer Intern*, berada di bawah naungan tim *Research & Development* seperti yang ditunjukkan pada Gambar 3.1. Kedudukan ini melibatkan tanggung jawab sebagai berikut:

- **Pembuatan Video Edukasi:** Merancang dan memproduksi video tutorial pengembangan game, mencakup riset topik, pembuatan skenario, dan penyusunan materi yang mudah dipahami.
- **Implementasi Kode untuk Demonstrasi:** Mengimplementasikan kode dalam proyek game untuk mendukung materi video, termasuk pengaturan objek, *scene*, dan interaksi dasar.
- **Dokumentasi Teknis dan Penulisan Laporan:** Mendokumentasikan setiap tahapan kerja, termasuk konsep game, implementasi teknis, dan evaluasi proses.



Gambar 3.1. Kedudukan penulis selama magang di PT DigiKidz Indonesia.
Sumber: Dokumen internal PT DigiKidz Indonesia, 2024.

3.1.2 Koordinasi

Penulis berkoordinasi langsung dengan koordinator *Product and Research Development Team*. Proses koordinasi dilakukan melalui:

- Laporan progres pekerjaan melalui platform *Google Drive*.
- Evaluasi hasil kerja oleh koordinator dengan umpan balik untuk revisi.
- Proses berulang dengan penyesuaian seiring pergantian proyek.

3.2 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke -	Pekerjaan yang dilakukan
1	Pengenalan Unity, pengaturan proyek, dan pengimporan aset untuk pengembangan game RPG 2D.
2	Membuat dunia game dengan Tile Palette, termasuk latar belakang dan elemen visual.
3	Implementasi karakter ke dalam game, termasuk gerakan dasar dan animasi.
4	Menambahkan interaksi karakter, seperti berbicara dengan NPC, dan sistem dialog dasar.
5	Pengembangan logika permainan dan pengujian fitur-fitur awal game RPG 2D.
6	Finalisasi game RPG 2D, termasuk pengemasan proyek dan dokumentasi teknis.
7	Pembuatan storytelling sederhana menggunakan Scratch untuk mengenalkan logika dasar pemrograman.
8	Pengembangan game pertarungan Marvel sederhana menggunakan Scratch.
9	Pembuatan game seperti Flappy Bird menggunakan Construct, dengan karakter yang diganti menjadi Superman.
10	Finalisasi game Scratch dan Construct, termasuk evaluasi proyek dan dokumentasi.
11	Pengenalan Unity untuk pengembangan game 3D, termasuk pembuatan objek game.
12	Implementasi elemen gameplay seperti kontrol karakter, sistem collision, dan pengaturan rintangan.
13	Penambahan logika permainan untuk skor, tingkat kesulitan, dan kondisi game over.
14	Finalisasi game 3D dengan pengujian lengkap, dokumentasi, dan persiapan presentasi.

3.3 Requirement Project

Sebelum memulai pengerjaan proyek pengembangan game, dilakukan identifikasi kebutuhan untuk memastikan hasil yang sesuai dengan tujuan dan audiens. Berikut adalah analisis kebutuhan yang dilakukan untuk setiap proyek.

3.3.1 Analisis Kebutuhan Umum

- **Target Audiens:** Anak-anak usia TK hingga SMA sesuai proyek yang dilakukan.
- **Platform:** Game akan dikembangkan untuk platform Windows dan Android, dengan pertimbangan fleksibilitas dan aksesibilitas.
- **Fitur Utama:**
  - Karakter yang dapat bergerak secara bebas dalam dunia 2D atau 3D.
  - Game dibangun menyesuaikan dengan tema *game* yang diberikan supervisi.
  - *Game* dibuat sederhana, dengan mempertimbangkan target audiens.
  - Latar belakang yang interaktif dengan elemen visual yang menarik.

3.4 Project 1: *Holiday Camp*

3.4.1 Analisis Kebutuhan

Sebelum mengembangkan *Project Holiday Camp*, dilakukan analisis kebutuhan untuk memastikan tujuan pembelajaran dan desain sesuai dengan target audiens. Berikut detail kebutuhannya:

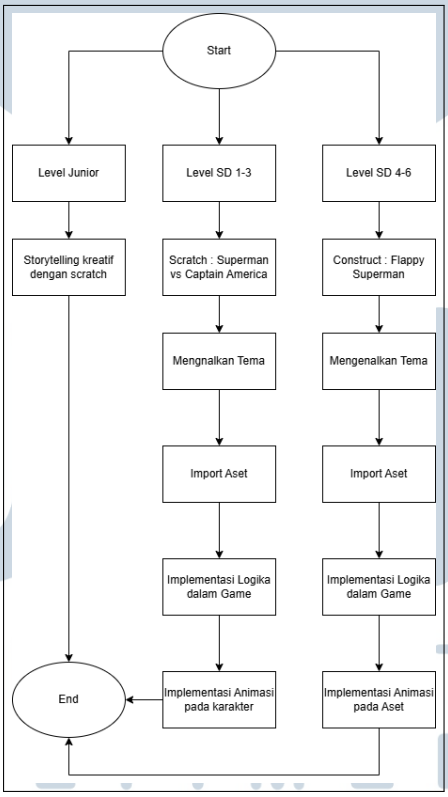
- **Tujuan:** Mengenalkan konsep dasar pemrograman kepada anak-anak usia TK hingga SD kelas 6 melalui pendekatan kreatif dan interaktif.
- **Metode:** Penggunaan *Scratch* untuk storytelling dan game sederhana serta *Construct* untuk pengembangan game tingkat lanjut.
- **Aset yang Dibutuhkan:** Detail aset yang dibutuhkan tercantum dalam Tabel 3.2. Tabel 3.2 menjelaskan sumber aset yang akan digunakan untuk pengembangan *game* dalam *project Holiday Camp*, seperti karakter, latar belakang, dan efek suara.

Tabel 3.2. Aset yang Dibutuhkan untuk *Holiday Camp*

Aset	**Sumber**
Karakter untuk storytelling	Scratch Library
Latar belakang cerita	Scratch Library
Musik latar interaktif	FreeSound.org
Efek suara	Zapsplat.com

3.4.2 Flowchart Alur Pembuatan *Holiday Camp*

Flowchart pada Gambar 3.2 menunjukkan alur pengembangan *Holiday Camp*. Alur dipecah menjadi 3 menyesuaikan dengan alur *project* yang dipecah menjadi 3 bagian. Penjelasan lebih mendalam mengenai pelaksanaan *project* akan dijelaskan penulis pada subbab 3.8.



Gambar 3.2. Flowchart Alur Game *Holiday Camp*

3.5 Project 2: 2D RPG Development

3.5.1 Analisis Kebutuhan

Proyek ini dirancang untuk mengajarkan logika pemrograman tingkat menengah kepada siswa SMA melalui pengembangan game RPG 2D. Berikut adalah detail kebutuhannya:

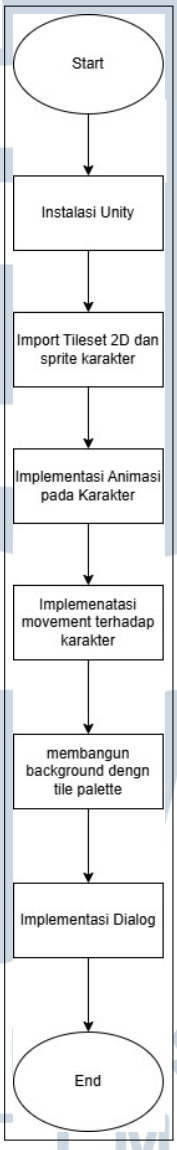
- **Tujuan:** Mengenalkan logika pemrograman tingkat menengah kepada siswa SMA melalui pengembangan game RPG 2D.
- **Platform:** Pengembangan menggunakan *Unity* dengan dukungan *Visual Studio*.
- **Aset yang Dibutuhkan:** Detail aset yang diperlukan ditampilkan dalam Tabel 3.3. Seluruh aset yang ditampilkan dalam 3.3 berasal dari sumber eksternal dan bukan merupakan karya penulis.

Tabel 3.3. Aset yang Dibutuhkan untuk 2D RPG Development

Aset	**Sumber**
Tileset dunia 2D	Itch.io (Cloud City Tileset)
Sprite karakter	Kenney.nl
Musik latar RPG	FreeSound.org

3.5.2 Flowchart Alur Pembuatan Game 2D RPG

Proses pengembangan game 2D RPG dirangkum dalam flowchart pada Gambar 3.3, yang mencakup tahapan persiapan hingga implementasi pada *project 2d RPG Game*. Penjelasan mengenai implementasi *project* terlampir dalam subbab 3.7.



Gambar 3.3. Flowchart Alur Game 2D RPG

3.6 Project 3: 3D Game Development

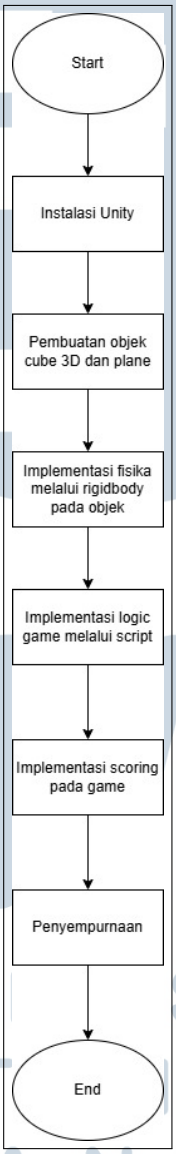
3.6.1 Analisis Kebutuhan

Proyek ini dirancang untuk mengajarkan logika pemrograman tingkat lanjut dan desain interaktif kepada siswa SMA melalui pengembangan *game* 3D. Berikut adalah kebutuhan yang telah diidentifikasi:

- **Tujuan:** Mengajarkan logika pemrograman tingkat lanjut dan desain interaktif kepada siswa SMA.
- **Platform:** Pengembangan menggunakan *Unity* dengan dukungan *Visual Studio*.
- **Aset yang Dibutuhkan:** Seluruh aset akan dikembangkan secara mandiri menggunakan *Unity*.

3.6.2 Flowchart Alur Pembuatan Game 3D

Proses pengembangan *game* 3D dirangkum dalam flowchart pada Gambar 3.4, yang mencakup tahapan mulai dari desain hingga pengujian. Alur pengembangan yang dilakukan menyerupai alur pada pengembangan *game 2d RPG*, dimana penulis menggunakan *Unity* sebagai *game engine*. Perbedaan kedua *project* terletak pada tema pengembangan dan aset yang digunakan.



Gambar 3.4. Flowchart Alur Pembuatan Game 3D

3.7 Detail Proyek Pengembangan Game 2D RPG

3.7.1 *Tools* yang digunakan untuk pengembangan *game*

Berikut adalah beberapa *tools* yang digunakan untuk mendukung pengembangan *game 2d RPG*

- ***Game Eengine Unity***
  Unity merupakan sebuah *software engine* yang digunakan untuk mengembangkan *game* dengan *genre casual, AR(Augmented Reality)* dan *VR(Virtual Reality)* [6]. Hasil game dari *Unity* bersifat *cross-platform*, yang berarti dapat dipublikasikan ke beberapa platform seperti *Windows, Android, Maxc, iOS, PS3,* dan lainnya. Penulis memilih menggunakan *Unity* sebagai tool pembelajaran *Game 2D* dikarenakan *Unity* memiliki tampilan antarmuka yang lebih mudah digunakan.

- ***Visual Studio***
  Visual Studio menyediakan berbagai alat bantu yang sangat berguna bagi pengembang C# dalam Unity, seperti:

  - ***IntelliSense***: Fitur yang memberikan saran otomatis untuk menyelesaikan kode, mempercepat penulisan dan meminimalisir kesalahan sintaks.
  - ***Debugging***: Memungkinkan pengembang untuk melacak dan memperbaiki bug dalam kode dengan menggunakan breakpoints dan pemeriksaan nilai variabel secara real-time.
  - **C*ode Navigation***: Fitur seperti go-to-definition, find references, dan refactoring membantu pengembang untuk dengan cepat memahami dan mengubah kode dalam proyek yang besar.
  - **Integrasi dengan *Unity***: Visual Studio terintegrasi langsung dengan Unity, sehingga setiap perubahan kode di Visual Studio dapat langsung diterapkan dan diuji di dalam Unity Editor.

3.7.2 Pengenalan Unity dan Konfigurasi *Game*

Langkah awal dalam proyek ini adalah mengenalkan Unity sebagai *game engine* yang dipilih karena fleksibilitasnya dan dukungan komunitas yang luas.Tahapan yang dilakukan untuk bagian ini meliputi:

- **Pengunduhan *Unity Hub* dan *Unity Editor***
  Untuk memastikan *user* dapat menggunakan *Unity* dengan baik, penulis membuat panduan pengunduhan *Unity* untuk setiap *Operating System* yaitu *Windows, Mac OS,*dan *Linux*.

- **Memulai proyek baru dalam *Unity* dengan pengaturan tipe *Game 2D***
  Pada tahap ini penulis membuat panduan membuat proyek baru dalam *Unity*, proyek baru ditujukan khusus untuk pengembangan *game* dua dimensi.

- **Penjelasan antarmuka *Unity*** Dalam antarmuka *Unity* terdapat 4 buah panel utama yang digunakan dalam pengembangan *game* :

  – Panel *Scene View* : Panel *Scene View* adalah tampilan editor di mana pengembang dapat melihat dan mengatur objek-objek dalam dunia game secara langsung.

  – Panel *Hierarchy* : Panel *Hierarchy* menunjukkan daftar semua objek dalam scene saat ini, memungkinkan pengembang untuk dengan mudah mengelola dan memilih objek.

  – Panel *Inspector* : Panel *Inspector* digunakan untuk menampilkan dan mengedit properti dari objek yang dipilih dalam *Hierarchy*, seperti transformasi, komponen, dan skrip.

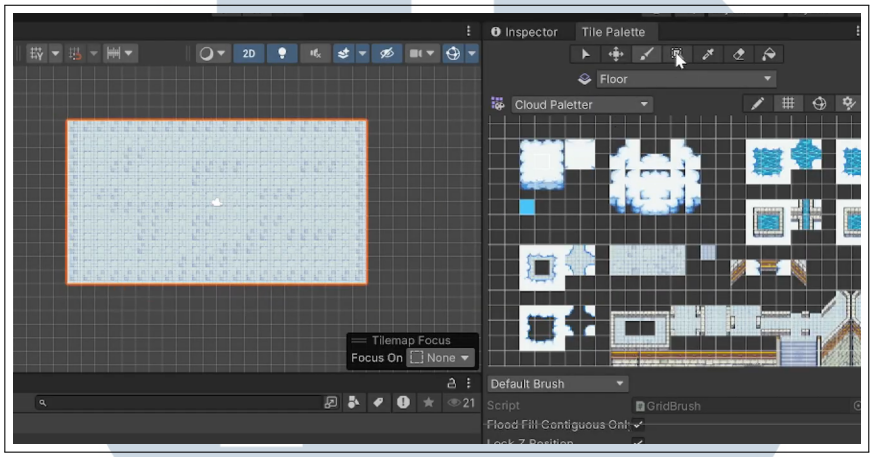
  – Panel *Assets* : Panel *Assets* adalah panel yang menampilkan semua file dan resource yang digunakan dalam proyek, termasuk tekstur, model, skrip, dan lainnya.

- **Pengimporan *tile set* dan penggunaan *Tile Palette* untuk pembuatan latar belakang dalam *game***

  Pengimporan *tile set* ke Unity dilakukan dengan menyeret file gambar ke dalam panel *Assets*. Setelah file berhasil diimpor, langkah selanjutnya adalah mengatur *Sprite Mode* menjadi *Multiple* dan memanfaatkan *Sprite Editor* untuk membagi gambar menjadi potongan-potongan kecil (*tiles*) berdasarkan ukuran grid yang sesuai, dalam konteks kali ini adalah 16x16 piksel. Potongan-potongan ini nantinya digunakan untuk menyusun elemen *background* dalam *game* secara modular.

  Pada tahap ini, penulis mengenalkan fitur bawaan Unity yang disebut *Tile Palette*. *Tile Palette* memungkinkan pengembang untuk menyusun elemen-elemen visual berbasis *tile* dengan mudah dan presisi. Dengan menggunakan

Tile Palette, pengembang dapat membuat, mengatur, dan menyimpan pola *tiles* untuk digunakan dalam berbagai bagian peta atau level seperti pada Gambar 3.5.



Gambar 3.5. Tampilan penggunaan *tile palette* dalam *Unity sprite sheet*.

Berikut adalah langkah-langkah yang perlu dilakukan dalam penggunaan *Tile Palette*:

- Membuat *palette* baru melalui menu *Window* *2D* *Tile Palette*, di mana pengguna dapat mengatur dan menyimpan potongan *tiles*.
- Menambahkan potongan *tiles* dari *tile set* yang telah diimpor ke dalam *palette*.
- Menggunakan alat *Brush* untuk menggambar elemen-elemen visual di dalam peta berdasarkan grid yang telah ditentukan.

Dengan *Tile Palette*, penulis dapat menyusun berbagai elemen latar belakang seperti lantai, dinding, dan elemen dekoratif lainnya. Pada bagian ini, penulis membuat latar belakang untuk game dengan mengatur *tile* pada layer yang sesuai menggunakan grid. Elemen-elemen latar belakang yang dibuat dengan *Tile Palette* juga mendukung penggunaan lapisan (*layers*), yang memudahkan pengelolaan elemen visual, seperti pemisahan antara elemen latar belakang dan elemen interaktif.

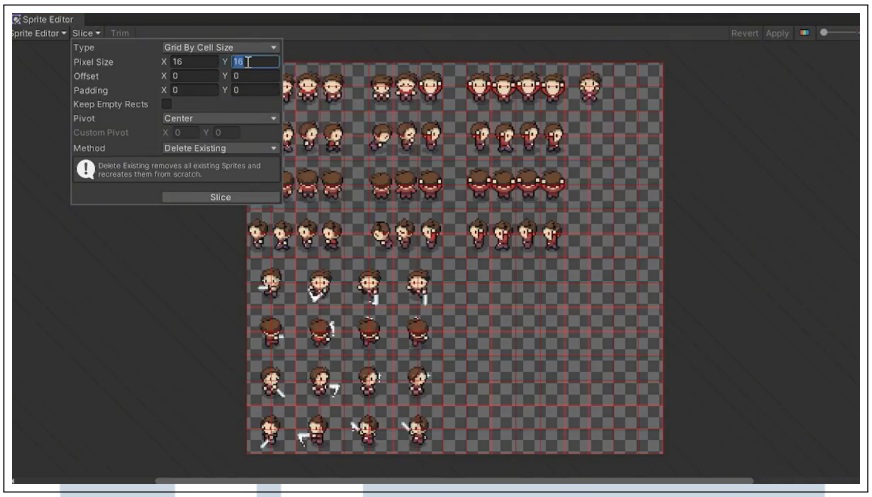
- **Pengimporan *sprite sheet* karakter dan memasukkan karakter ke dalam *scene***

  Setelah menyelesaikan pengaturan latar belakang menggunakan *Tile Palette*, langkah selanjutnya adalah mengimpor *sprite sheet* karakter ke dalam Unity. Proses ini dimulai dengan menyeret file *sprite sheet* ke dalam panel *Assets*. File *sprite sheet* ini berisi kumpulan gambar karakter dalam berbagai posisi atau animasi.

  Langkah pertama dalam pengaturan *sprite sheet* adalah dengan mengubah properti *Sprite Mode* menjadi *Multiple* di panel *Inspector*. Selanjutnya, penulis menggunakan *Sprite Editor* untuk memotong gambar menjadi beberapa *sprite* individu yang mewakili posisi atau gerakan karakter. Dalam kasus ini, setiap potongan *sprite* merepresentasikan animasi tertentu seperti berjalan, berdiri diam, atau melompat.

  Setelah potongan *sprite* selesai, *sprite* karakter ditambahkan ke dalam *scene* sebagai objek. Langkah-langkah untuk menambahkan karakter ke dalam *scene* adalah sebagai berikut:

  – Memilih salah satu potongan *sprite* dari panel *Assets* dan menyeretnya ke panel *Scene View*. Potongan *sprite* tersebut otomatis menjadi sebuah *GameObject* dalam Unity.
  – Mengatur posisi karakter di dalam *scene* menggunakan *Transform Tool*, sehingga karakter berada di lokasi yang sesuai dalam lingkungan *game*.
  – Memberikan nama pada objek karakter di panel *Hierarchy* untuk memudahkan identifikasi dan pengelolaan objek di dalam *scene*.



Gambar 3.6. Tampilan *Sprite Editor* di Unity untuk memotong *sprite sheet* karakter.

3.7.3 Implementasi Gerakan dan Animasi pada Karakter

Gerakan karakter dalam game dikembangkan menggunakan bahasa pemrograman C#, yang memungkinkan pengontrolan posisi karakter di dunia 2D dengan menangani input dari pengguna. Pada tahap ini, karakter diprogram untuk merespons perintah yang diberikan melalui tombol arah atau WASD pada keyboard. Selain itu, animasi karakter juga ditambahkan agar gerakan terlihat lebih realistis dan dinamis. Animasi diimplementasikan dengan menggunakan panel *Animation* dan *Animator*, yang memungkinkan pengaturan animasi berdasarkan input pengguna. Cuplikan implementasi dapat dilihat pada Gambar 3.7. Langkah-langkah implementasinya adalah sebagai berikut:

1. **Menambahkan komponen *Rigidbody2D***: Komponen ini digunakan untuk menangani fisika dasar pada objek 2D, seperti gravitasi dan kolisi. Dengan menambahkan *Rigidbody2D* pada karakter, Unity dapat mengelola gerakan karakter secara fisik dan realistis. Meskipun pada kasus ini kita memanipulasi posisi secara langsung, *Rigidbody2D* tetap dibutuhkan untuk mendukung interaksi fisika di masa mendatang. *Rigidbody2D* dapat dimasukkan melalui panel *Inspector* dari objek pemain.
2. **Menggunakan fungsi `Input.GetAxis`**: Fungsi ini digunakan untuk menangkap input dari pengguna. Unity menyediakan `Input.GetAxis("Horizontal")` dan `Input.GetAxis("Vertical")` untuk mendapatkan nilai input dari tombol arah atau WASD. Fungsi ini

mengembalikan nilai antara -1 dan 1, mencerminkan arah gerakan horizontal (kiri/kanan) dan vertikal (atas/bawah).

3. **Mengatur kecepatan gerakan**: Kecepatan gerakan karakter disesuaikan melalui variabel `moveSpeed` yang didefinisikan dalam skrip dan dapat diatur dari *Inspector* Unity. Hal ini memudahkan pengembang untuk mengatur kecepatan karakter tanpa mengubah kode sumber.

4. **Menambahkan Animasi Karakter Menggunakan Panel *Animation***: Animasi gerakan karakter dibuat dengan memanfaatkan *sprite sheet* yang telah diimpor sebelumnya. Langkah-langkah untuk menambahkan animasi adalah:

    - Pilih karakter di dalam *scene*, lalu buka panel *Animation*.
    - Buat animasi baru, seperti *Idle*, *Walk Up*, *Walk Down*, dan *Walk Side*.
    - Tambahkan potongan-potongan *sprite* dari *sprite sheet* ke dalam *Timeline* animasi.
    - Atur durasi animasi pada *Timeline* untuk menciptakan transisi yang mulus di antara frame-frame *sprite*.

5. **Menggunakan Panel *Animator* untuk Menghubungkan Animasi dengan Input**: Panel *Animator* digunakan untuk membuat transisi di antara animasi berdasarkan input pemain. Langkah-langkahnya adalah:

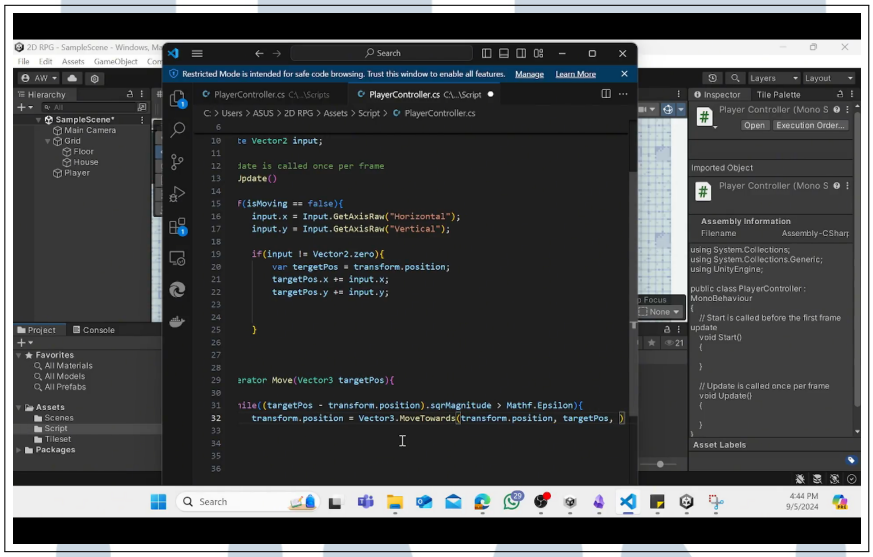
    - Buka panel *Animator* dan tambahkan setiap animasi (*Idle*, *Walk Up*, *Walk Down*, dan *Walk Side*) ke dalam *State Machine*.
    - Buat parameter *Float* untuk arah gerakan, seperti `MoveX` dan `MoveY`.
    - Hubungkan parameter ini ke kondisi transisi di antara animasi. Contohnya, jika `MoveX > 0`, karakter akan menggunakan animasi berjalan ke kanan.

6. **Menyesuaikan Skrip `PlayerController` untuk Animasi**: Skrip `PlayerController` yang sebelumnya hanya menangani gerakan karakter sekarang diperbarui untuk mengontrol parameter di *Animator*. Cuplikan kode dapat dilihat pada Gambar 3.8. Penyesuaian utama adalah:

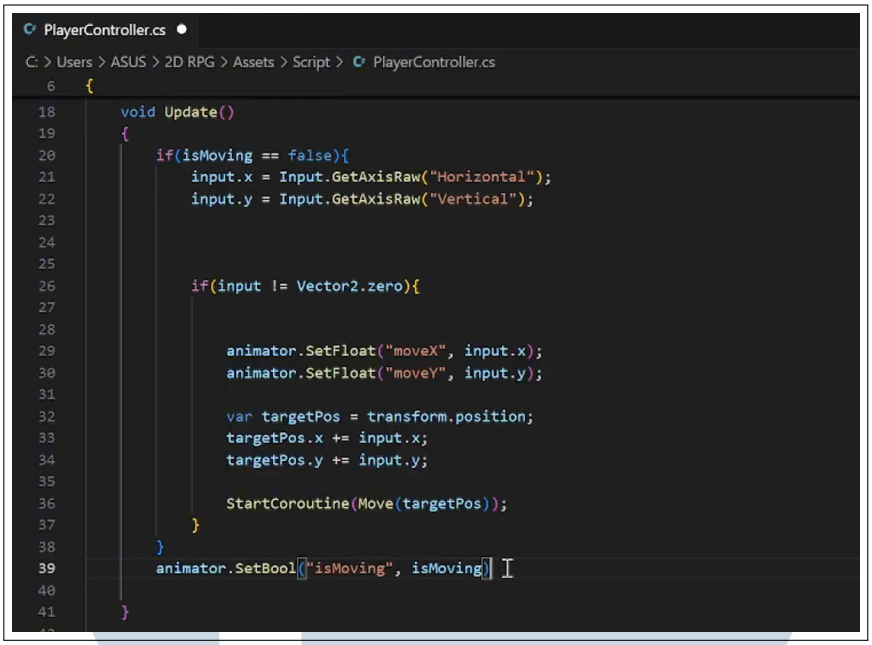
    - Menambahkan referensi ke `Animator` di dalam skrip.

- Mengatur nilai parameter `MoveX` dan `MoveY` berdasarkan nilai input dari `Input.GetAxis`.
- Mengatur kondisi animasi berdasarkan kecepatan dan arah gerakan karakter.

Implementasi ini memberikan kontrol penuh terhadap gerakan dan animasi karakter, memungkinkan transisi animasi yang halus sesuai dengan input pengguna. Karakter dapat bergerak secara dinamis ke kiri, kanan, atas, dan bawah dengan animasi yang sesuai, menciptakan pengalaman bermain yang lebih imersif.



Gambar 3.7. Implementasi skrip untuk gerakan dan animasi karakter di Unity.



Gambar 3.8. Kode untuk mengontrol gerakan dan animasi karakter menggunakan `PlayerController`.

3.7.4 Sistem Dialog Dinamis

Sistem dialog dinamis dirancang untuk memungkinkan pemain berinteraksi dengan NPC (Non-Playable Character) secara lebih fleksibel, dengan alur percakapan yang dapat menyesuaikan berdasarkan tindakan atau pilihan pemain. Implementasi sistem ini dilakukan dalam beberapa tahap, seperti dijelaskan berikut:

1. **Pembuatan Skrip Sistem Interaksi dan Dialog**
   Langkah awal adalah membuat skrip yang mengatur interaksi antara pemain dan NPC. Skrip ini mendeteksi saat pemain berada dalam jarak tertentu dari NPC dan menekan tombol interaksi (misalnya tombol `E`), sehingga memicu dialog yang relevan. Skrip ini juga bertanggung jawab untuk mengelola alur percakapan.

2. **Pengujian dengan `Debug.Log`**
   Sebelum melanjutkan ke tahap implementasi visual, alur dialog diuji menggunakan `Debug.Log` untuk memberikan umpan balik real-time melalui konsol Unity. Dengan cara ini, pengembang dapat memastikan bahwa logika interaksi dan alur dialog telah berjalan sesuai harapan. Sebagai contoh:

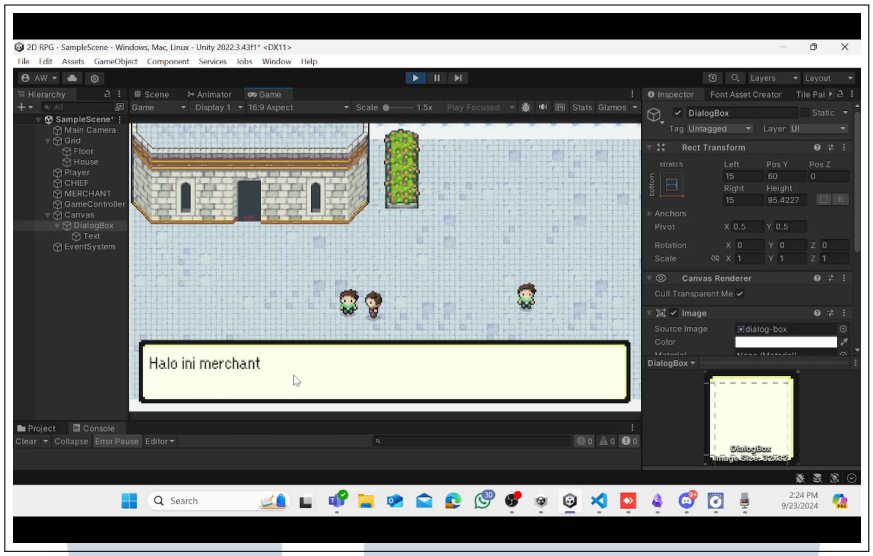
```
Debug.Log("Pemain berinteraksi dengan NPC.
Menampilkan dialog...");
```

3. **Implementasi Elemen Visual Dialog Box**
   Setelah logika dialog terverifikasi, tahap selanjutnya adalah mengimplementasikan antarmuka pengguna (UI) berupa kotak dialog menggunakan *Canvas* di Unity. Elemen visual ini mencakup kotak dialog dan teks yang akan ditampilkan di layar. Proses ini memastikan tampilan dialog sesuai dengan alur percakapan yang telah diprogram.

4. **Penambahan Efek Teks Berjalan dengan `Coroutine`**
   Untuk meningkatkan pengalaman pengguna, sistem ini menampilkan teks dialog secara bertahap menggunakan `Coroutine`. Dengan pendekatan ini, setiap karakter dalam teks dialog muncul satu per satu dengan jeda tertentu, memberikan kesan bahwa NPC sedang berbicara. Selain itu, fitur ini memungkinkan penambahan efek transisi antar dialog.

5. **Logika Dinamis Berdasarkan Pilihan Pemain**
   Tahap akhir adalah menambahkan logika yang memungkinkan dialog berubah sesuai pilihan pemain. Opsi dialog yang dipilih oleh pemain akan memengaruhi respons NPC dan percakapan selanjutnya. Untuk mencapai ini, digunakan logika kondisi seperti `if-else` atau `switch-case`, sehingga percakapan terasa lebih dinamis dan interaktif.

Pada tahap pengembangan ini, meskipun elemen UI dialog box belum sepenuhnya diterapkan, logika dasar telah diuji dengan `Debug.Log` untuk memastikan sistem berfungsi sesuai desain. Sistem ini memberikan fleksibilitas bagi pengembang untuk memverifikasi alur dialog dan memastikan respons NPC sesuai dengan input pemain.



Gambar 3.9. Ilustrasi sistem dialog dinamis dengan efek teks berjalan.

3.8 Detail Proyek Perancangan Program Holiday Camp

3.8.1 Pendahuluan Program Holiday Camp

Program Holiday Camp merupakan kegiatan akhir tahun yang bertujuan untuk mengajarkan konsep dasar pemrograman kepada peserta dari berbagai tingkat pendidikan. Program ini menggunakan pendekatan kreatif dan interaktif untuk memperkenalkan logika pemrograman dan pengembangan game sederhana. Fokus utama diarahkan pada pengembangan game menggunakan alat seperti Scratch dan Construct.

3.8.2 Pengenalan Mengenai *Scratch*

Scratch adalah platform pemrograman visual yang dirancang untuk memudahkan pembelajaran pemrograman bagi pemula. Dengan sistem blok perintah, pengguna dapat membuat cerita interaktif, animasi, dan game sederhana tanpa menulis kode secara langsung. Scratch memungkinkan peserta memahami konsep logika pemrograman seperti alur kontrol, variabel, dan perulangan secara intuitif.

3.8.3 Pengenalan Mengenai *Construct*

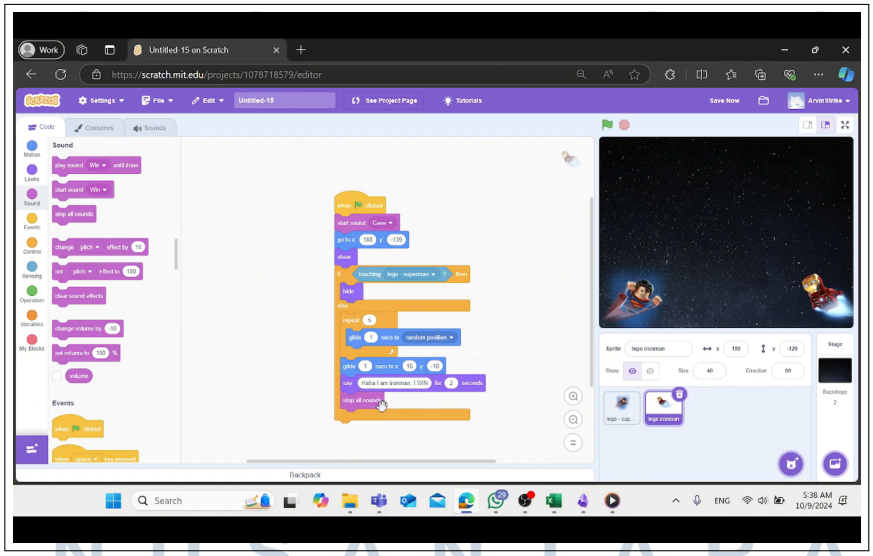
Construct adalah perangkat lunak berbasis HTML5 untuk pengembangan game 2D yang memungkinkan pengguna membuat game tanpa memerlukan pengetahuan pemrograman mendalam. Dengan antarmuka drag-and-drop serta logika berbasis event, Construct cocok bagi peserta tingkat lanjut yang ingin mempelajari mekanisme dasar pengembangan game.

3.8.4 *Game Project* untuk Level Junior: Storytelling dengan Scratch

Peserta level junior diajarkan konsep dasar pemrograman melalui pembuatan cerita interaktif sederhana di Scratch. Proses pembelajaran meliputi:

- Mengenalkan fungsi dasar Scratch, seperti pemilihan *sprite* dan blok perintah sederhana.
- Membimbing peserta merancang cerita dengan dialog dan animasi.
- Mengevaluasi kreativitas dan pemahaman peserta terhadap konsep pemrograman dasar.

Antarmuka Scratch untuk kegiatan ini ditunjukkan pada Gambar 3.10.



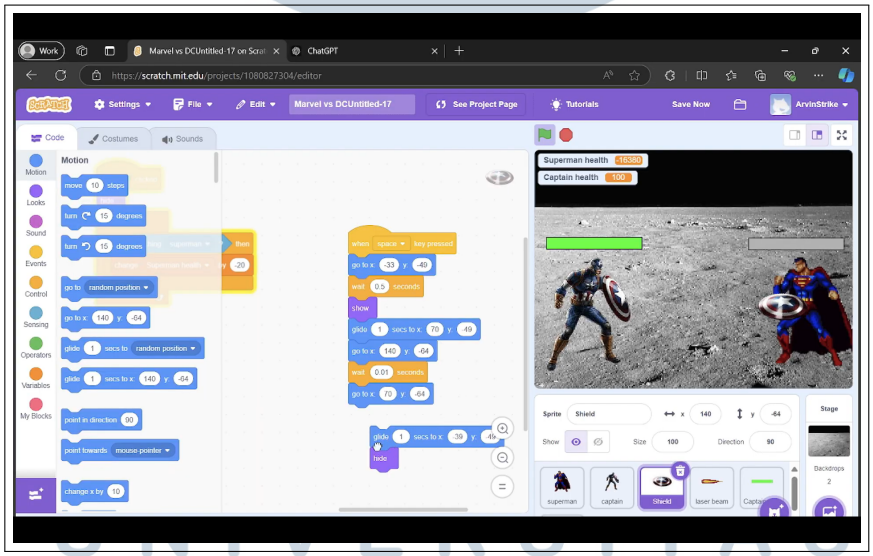
Gambar 3.10. Pembuatan cerita interaktif menggunakan Scratch.

3.8.5 Level SD Kelas 1-3: Game Scratch - Superman vs Captain America

Peserta SD kelas 1-3 belajar mengembangkan game sederhana menggunakan Scratch dengan fokus pada logika dan animasi dasar. Proyek ini mencakup:

- **Mengenalkan Karakter:** Peserta memilih karakter superhero seperti Superman dan Captain America dari pustaka *sprite*.
- **Membuat Animasi:** Memberikan gerakan dasar seperti berjalan atau melompat pada karakter.
- **Desain Rintangan:** Menambahkan rintangan sederhana, seperti batu yang bergerak, untuk menciptakan tantangan.
- **Sistem Skor:** Membuat logika perhitungan skor berdasarkan waktu bertahan atau poin yang dikumpulkan.

Hasil akhir pengembangan game ini ditampilkan pada Gambar 3.11.



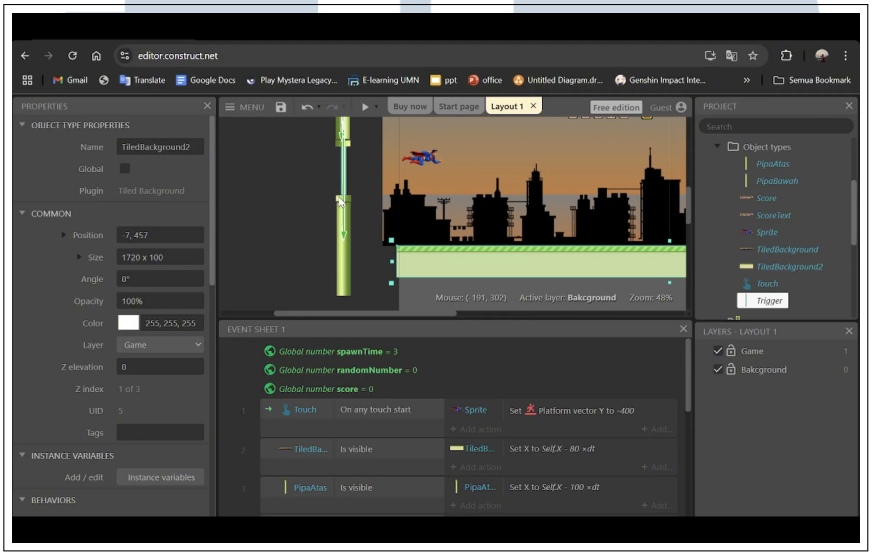
Gambar 3.11. Pengembangan game Superman vs Captain America dengan Scratch.

3.8.6 Level SD Kelas 4-6: Game Flappy Bird dengan Construct

Peserta SD kelas 4-6 mempelajari mekanisme dasar pengembangan game menggunakan Construct dengan fokus pada pembuatan game mirip Flappy Bird. Tahapan pembelajaran meliputi:

- Mengimpor *sprite* dan latar belakang permainan.
- Mengembangkan mekanisme kontrol untuk melompat dengan menekan tombol.
- Menambahkan sistem skor berdasarkan jumlah rintangan yang berhasil dilewati.

Hasil pengembangan game ini ditunjukkan pada Gambar 3.12.



Gambar 3.12. Hasil pengembangan game Flappy Bird dengan Construct.

3.9 Kendala dan Solusi yang Ditemukan

3.9.1 Kendala

Berikut beberapa kendala yang dihadapi selama pelaksanaan proses magang:

1. Kendala dalam komunikasi selama WFH penuh Pelaksanaan kerja dari rumah secara penuh (WFH) mengakibatkan keterbatasan komunikasi, terutama dalam diskusi lintas penulis dan koordinasi dengan supervisor. Hal ini memperlambat proses penyelesaian tugas serta menyebabkan kesalahpahaman dalam interpretasi tugas yang diberikan.
2. Kesulitan dalam riset awal terkait proyek pengembangan game Pada tahap inisialisasi awal, terdapat kendala dalam memahami konteks proyek game

yang diberikan, termasuk menentukan alat, platform, serta mekanika game yang relevan untuk materi pembelajaran. Hal ini diperburuk oleh kurangnya sumber referensi spesifik mengenai kebutuhan dan standar game edukasi yang sesuai dengan audiens anak-anak.

3. Kurangnya pemahaman terhadap proses bisnis di PT Digikidz Indonesia Sebagai pendatang baru, memahami struktur organisasi, alur kerja, dan sistem operasional PT Digikidz Indonesia menjadi tantangan tersendiri. Hal ini menyebabkan keterlambatan dalam menyelaraskan tugas dengan tujuan perusahaan, terutama dalam mengintegrasikan proyek dengan kebutuhan pembelajaran di tempat les.

3.9.2 Solusi

Berikut beberapa solusi dari kendala yang dihadapi selama pelaksanaan proses magang:

1. Mengoptimalkan penggunaan alat komunikasi daring seperti Zoom, Google Meet, dan Slack secara lebih terstruktur, termasuk menetapkan jadwal rapat rutin dan menciptakan dokumentasi tugas secara rinci di platform kolaborasi seperti Google Drive atau Notion.
2. Melakukan kolaborasi dengan mentor atau senior untuk memperoleh panduan awal terkait standar pengembangan game edukasi. Selain itu, memanfaatkan forum daring seperti Unity Community, Scratch Forums, dan Construct Forums untuk mempercepat proses pembelajaran serta mencari inspirasi dari proyek serupa yang sudah ada.
3. Mengadakan sesi orientasi atau diskusi dengan supervisi untuk mempelajari alur bisnis perusahaan secara menyeluruh.