***FINAL PROJECT* PBO *SAMURAI TURN-BASED ROLE-PLAYING GAME***



Dosen Pengampu:

I Made Suartana, S.Kom., M.Kom.

Disusun oleh:

Kelompok 2

Rayhan Ramadhani Hendra Atmadja 23051204075

Adriano Emmanuel 23051204082

Cornelius Louis Nathan 23051204085

PROGRAM STUDI (S1) TEKNIK INFORMATIKA  
 FAKULTAS TEKNIK  
 UNIVERSITAS NEGERI SURABAYA  
 2024

# KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan Yang Maha Esa. Karena atas limpahan rahmat-Nya kami dapat menyelesaikan laporan ini yang berjudul “*FINAL PROJECT* PBO *SAMURAI TURN-BASED ROLE-PLAYING GAME*” dengan tepat waktu tanpa ada halangan yang dan sesuai dengan harapan.

Adapun tujuan dari penulisan laporan ini adalah untuk memahami dan memperdalam bagaimana pengimplementasian PBO dengan menggunakkan bahasa pemrograman Python menggunakkan *library* *pygame*. Pembuatan laporan ini juga menjadi tugas yang mengisi nilai Ujian Akhir Semester dari Mata Kuliah Pemrograman Berbasis Objek.

Penyusunan laporan ini tidak lepas dari bantuan dan dukungan. Sehingga, kami menyampaikan terima kasih kepada pihak yang sudah mendukung pembuatan laporan ini. Terutama kepada Bapak I Made Suartana, S.Kom., M.Kom., selaku dosen pengampu mata kuliah Pemrograman Berbasis Objek yang telah membantu memberikan arahan dan pemahaman dalam penyusunan laporan ini.

Kami menyadari bahwa dalam penyusunan laporan ini masih banyak kekurangan karena keterbatasan kami. Maka dari itu kami mengharapkan kritik dan saran dari pembaca untuk menyempurnakan laporan ini. Semoga apa yang ditulis dapat bermanfaat bagi semua pihak yang membutuhkan.

Surabaya, 10 Desember 2024

                 Tertanda,

Kelompok 2

# DAFTAR ISI

[KATA PENGANTAR i](#_Toc187356588)

[DAFTAR ISI ii](#_Toc187356589)

[BAB I PENDAHULUAN 1](#_Toc187356591)

[1.1  Latar Belakang 1](#_Toc187356592)

[1.2. Rumusan Masalah 1](#_Toc187356593)

[1.3. Batasan Masalah 2](#_Toc187356594)

[1.4. Tujuan 2](#_Toc187356595)

[1.5. Manfaat 2](#_Toc187356596)

[BAB II TINJAUAN PUSTAKA 3](#_Toc187356598)

[2.1  Turn-Based Role-Playing Games 3](#_Toc187356599)

[2.2  Pygame 3](#_Toc187356600)

[BAB III PENGAPLIKASIAN 4](#_Toc187356602)

[3.1 Konsep Pembuatan 4](#_Toc187356603)

[3.1.1 Rancangan *Class Diagram* 4](#_Toc187356604)

[3.1.2 *Library* dan Asetyang Digunakan 11](#_Toc187356605)

[3.2 Program Utama 24](#_Toc187356606)

[3.3 Dokumentasi Program 29](#_Toc187356607)

[3.3.1 Halaman awal permainan 29](#_Toc187356608)

[3.3.2 Tampilan Program 30](#_Toc187356609)

[3.4 Implementasi PBO 38](#_Toc187356610)

[BAB IV PENUTUP 42](#_Toc187356612)

[4.1 Kesimpulan 42](#_Toc187356613)

[4.2 Saran 43](#_Toc187356614)

# BAB I

# PENDAHULUAN

## 1.1  Latar Belakang

Industri game terus mengalami perkembangan dengan munculnya berbagai genre dan konsep yang selalu berbeda. Salah satu genre tersebut adalah *turn-base role-playing game (Turn-based RPG)*. *Turn-based RPG* adalah genre *game* yang berfokus pada pengalamanstrategis yang dimana pemain, sebagai karakter utama, secara bergiliran mengambil tindakan, seperti menyerang, bertahan, menggunakkan kemampuan, yang kemudian musuh merespon pada gilirannya. Banyak sekali *game-game* besar yang berhasil mempopulerkan *genre* *Turn-Based role-playing*, seperti *Final Fantasy*, *Pokemon*, *Persona, Vision of Mana*.

Untuk menciptakan suatu *game* sederhana, python menjadi salah satu bahasa pemrograman yang paling banyak digunakan dan direkomendasikan karena fleksibilitas, ketersediaan *library*, dan komunitas yang luas. Salah satu *library* yang digunakan untuk mendukung pengembangan *game* adalah *pygame*. *Pygame* menyediakan beberapa fitur untuk mengatur grafis, animasi, dan suara yang sehingga memudahkan proses pembuatan *game 2D*.

Tema samurai memiliki daya tarik kuat karena samurai mencerminkan nilai-nilai  kedisiplinan, kehormatan, ketenangan dalam mengambil keputusan, sesuai dengan mekanisme *turn-based RPG* yang dibawakan. Kombinasi antara samurai dan sistem *turn-based* menciptakan pengalaman bermain yang menegangkan, karena setiap tindakan yang salah tentunya akan mengakibatkan efek domino pada ronde selanjutnya.

Dari pembahasan di atas, laporan ini bertujuan untuk mengeksplorasi pengembangan *turn-based RPG* menggunakkan pythondan *pygame* sebagai dasar perkembangannya. Dipadukan dengan tema samurai, proyek ini dapat memberikan gambaran bagaimana seorang samurai perlu berhati-hati dalam mengambil keputusan.

## 1.2. Rumusan Masalah

1. Apa saja langkah-langkah proses pembuatan *Samurai Turn-Based Role-Playing Game*?
2. Bagaimana hasil implementasi *Samurai Turn-Based Role-Playing Game*?
3. Apa saja implementasi PBO dari *Samurai Turn-Based Role-Playing Game*?

## 1.3. Batasan Masalah

1. Makalah ini membahas rancangan *class diagram*, *library*, *framework* yang digunakan, fitur program serta dokumentasi saat program dijalankan.
2. Makalah ini juga membahas apa saja implementasi PBO dari *Samurai Turn-Based Game*.

## 1.4. Tujuan

1. Mengetahui langkah-langkah proses pembuatan *Samurai Turn-Based Role-Playing Game*.
2. Mengetahui hasil implementasi *Samurai Turn-Based Role-Playing Game*.
3. Mengetahui cara kerja algoritma dari *Samurai Turn-Based Role-Playing Game*.
4. Mengetahui hasil implementasi *Samurai Turn-Based Role-Playing Game*

## 1.5. Manfaat

1. Membantu pembaca memahami  proses pembuatan *Samurai Turn-Based Role-Playing Game*.
2. Menjadi referensi bagi pemula yang ingin mempelajari *pygame* dari python.
3. Membantu pembaca memahami  bagaimana implementasi PBO pada *Samurai Turn-Based Role-Playing Game*.

# BAB II

# TINJAUAN PUSTAKA

## 2.1  Turn-Based Role-Playing Games

*Turn-Based Role-Playing Games* sebenarnya gabungan dari dua genre yang berbeda, yaitu *turn-based* dan *role-playing games*. *Turn-based* secara harfiah artinya berbasis giliran, *role-playing games* adalah *game* yang pemainnya memerankan suatu karakter dalam sebuah cerita fantasi (Setiawan, dkk 2018 : 194). Pemain akan terlibat dalam suatu jalan cerita yang panjang dan turut berperan dalam *character development* dari karakter yang dimainkan.

*Turn-based Role-Playing games* sebenarnya bukan genre yang baru, karena sudah banyak *game* besar yang telah mengusung genre ini, seperti *Final Fantasy* yang dirilis pada tahun 1987 dengan masih menggunakkan grafik *pixelated*. *Revelations : Persona* atau yang dikenal di Jepang sebagai *Megami Inboku Persona* juga ikut meramaikan *genre* ini di tahun 1996.

## 2.2  Pygame

*Pygame* adalah *library* python yang berisi kumpulan fungsi untuk membantu perkembangan aplikasi *video game*. *Pygame* dirancang secara *cross-platform*, sehingga bisa dijalankan pada Windows, Linux, MacOS, ataupun sistem operasi lainnya, dan ditulis dalam bahasa pemrograman Python, C, Cython, Assembly.

*Pygame* menggunakkan *Simple DirectMedia Layer* (SDL) *library* yang memungkinkan untuk mengembangkan aplikasi *video game* secara *real-time* tanpa harus berurusan dengan *low-level mechanism* dari bahasa pemrograman C yang cukup membingungkan. Fungsi-fungsi paling berat pada *video game* yang dibuat pada *pygame* dapat diambil alih oleh *library SDL*, sedangkan *game logic* dapat menggunakkan bahasa pemrograman python.

# BAB III

# PENGAPLIKASIAN

## 3.1 Konsep Pembuatan

Hal pertama yang kami lakukan adalah menentukan judul *game*. Judul *game* yang dibawakan haruslah menarik dan mencerminkan tema *game* yang diusung. Karena kami terinspirasi dari *game “Ghost of Tsushima”* yang bergenre *Action-Adventure* dan sekaligus juga membawa tema samurai. Kami memutuskan untuk memilih judul yaitu “*Tale of Samurai*”, yang artinya adalah Kisah Samurai.

Saat ini, kami merancang *game* untuk berjalan dalam satu kali *match*. Setelah karakter utama berhasil mengalahkan semua musuh yang ada, maka *game* akan berakhir. Pemain dapat memilih opsi untuk melakukan *restart* untuk bermain kembali atau mengakhiri program.

Untuk karakter yang digunakan, sesuai dengan tema yang kami usung, kami menggunakkan samurai sebagai karakter utama. Sedangkan musuh-musuh yang dipilih, awalnya kami memilih menggunakkan ninja dan beberapa karakter umum lain. Namun untuk memberi pengalaman bermain yang berbeda, kami memutuskan untuk menggunakkan *yokai* - makhluk supranatural dalam mitologi Jepang - sebagai musuh dari karakter utama. Kami merasa penggunaan *yokai* masih relevan dengan samurai karena keduanya sama-sama berasal dari tradisi dan asal yang sama.

Mengenai *design game* yang digunakan, kami memilih menggunakkan *design pixelated*. Alasan utama kami menggunakkan *design pixelated* karena banyaknya aset yang tersedia secara gratis di internet. Dengan menggunakkan *design pixelated*, kami dengan mudah dapat memilih dan mengunduh aset yang dibutuhkan, tanpa perlu membuat asetnya dari awal.

Dalam proses pembuatan dari *Tale of Samurai*, kami membuat rancangan *class* *diagram* yang berisi *class*, *function*, serta relasi antar *class*. Tak lupa kami juga menentukan *library* dan aset yang akan digunakan untuk pembuatan *game* kami.

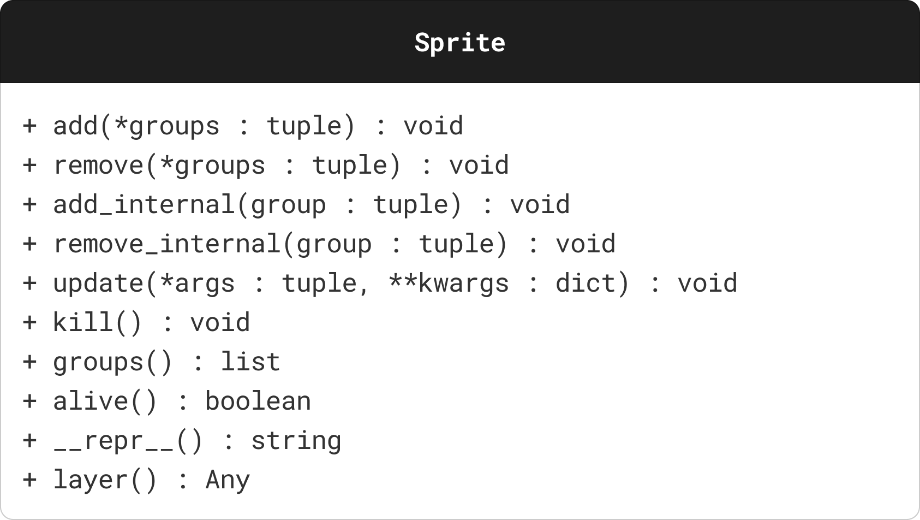
### 3.1.1 Rancangan *Class Diagram*

Suatu aplikasi yang mengusung *Object-Oriented Programming*  harus memiliki *class* sebagai *blueprint* dan *object* sebagai *instance* yang dibuat dari *class* tersebut. Maka dari itu, kami berdiskusi menentukan *class-class* apa saja yang dibutuhkan pada perkembangan *game* kami. Berikut adalah hasil pembahasan kami :

#### *Sprite*

*Sprite* adalah *class* yang digunakan untuk merepresentasikan objek *game* yang dapat di-*render* dan di-*update* di layar. *Class Sprite* membantu mengelola objek yang memiliki banyak *frame*, seperti karakter, musuh, proyektil, efek, dan item. Maka dari itu, *class Sprite* akan digunakan pada *game* ini dan akan dihubungkan dengan beberapa *class* lain agar animasi dan pergerakan objek lebih terstruktur dan dinamis.

Terdapat beberapa fungsi penting yang tersedia pada *Sprite*, seperti *def update()* untuk memperbarui posisi atau *frame* objek, *def draw()* untuk menampilkan objek pada layar. Berikut adalah diagram dari *class Sprite*:



***Gambar 3.1.1.1*** *Diagram dari Class Sprite (Sumber : Dokumentasi Penulis)*

#### *Character*

Langkah pertama yang kami lakukan adalah membuat *class* *character*. Suatu *character* tentunya harus mempunyai *name*, *hp* (*hitpoints* atau darah) *, max\_hp*, dan *attack* (serangan). Awalnya, kami berencana untuk membuat *class MC* dan *class enemy* dibuat secara terpisah. Namun, untuk menghindari penulisan kode berulang, kami memutuskan untuk menggabungkan dua *class* tersebut dalam satu *class Character*.

 Kami menambahkan *charm* atau yang lebih dikenal sebagai *buff* kepada *MC (Main Character*) untuk tujuan menyeimbangkan permainan, *charm* tersebut terdiri atas *refill\_health* dan *double\_damage*, sedangkan karakter musuh tidak memiliki *charm*. *Character* juga ditambahkan variabel *attack\_target*, *is\_attacking* danfungsi *attack()* dan *take\_damage()* agar *character* dapat saling menyerang satu dengan yang lain.

Setelah membuat *class* *character*, kami menghubungkan *class* *character* dengan *class sprite* yang merupakan *class* bawaan dari *pygame*. Tujuannya adalah agar *character* dapat dianimasikan dan diterapkan dalam *game loop* *pygame*. Secara tidak langsung, *class character* mewarisi beberapa fungsi dari *class Sprite* (*inheritance*). Selain itu, untuk mendukung animasi dan gerak *character*, kami menambahkan beberapa variabel seperti *x, y, animation\_list*, *animation\_index*, *alpha*, *current\_time, alpha, attack\_start\_time* dan beberapa fungsi seperti *def draw()*, *def animation(), def idle(), def update()*.

Setelah menambahkan beberapa variabel, fungsi dan melakukan beberapa penyesuaian, berikut adalah diagram dari *class Character* :



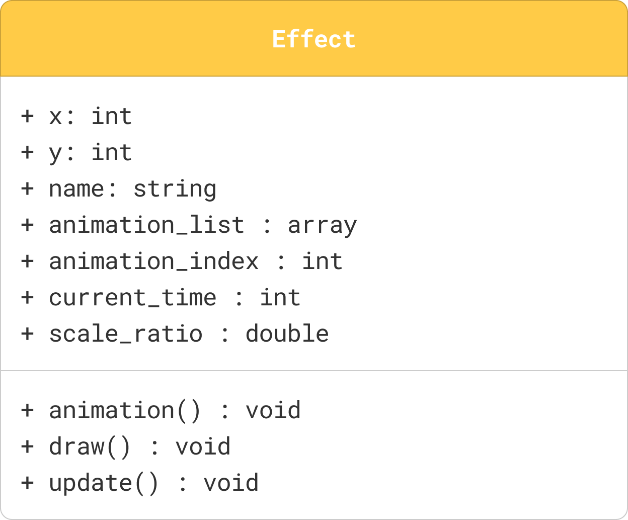
***Gambar 3.1.1.2*** *Diagram dari Class Character* *(Sumber : Dokumentasi Penulis)*

#### *Effect*

Sesuai namanya, *effect* merupakan tampilan visual yang muncul ketika suatu *character* melakukan serangan ke *character* lain. Maka, *class effect* dibuat untuk menghindari penulisan kode berulang. *Class effect* memiliki variabel *x* dan *y* untuk menetapkan posisi dari *effect* yang akan dibuat*, name* untuk menyimpan nama efek. *Class effect* merupakan agregasi dari *class character* karena *character* memiliki efek yang dapat ditampilkan.

Sama seperti *class* *character*, kami menghubungkan *class* *effect* dengan *class sprite* yang merupakan *class* bawaan dari *pygame*. Tujuannya adalah agar *effect* dapat dianimasikan dan diterapkan dalam *game loop* *pygame*. Secara tidak langsung, *class effect* mewarisi beberapa fungsi dari *class Sprite* (*inheritance*). Selain itu, untuk mendukung animasi dari *class effect*, kami menambahkan beberapa variabel seperti *x, y, animation\_list*, *animation\_index*, *current\_time*, *scale\_ratio* dan beberapa fungsi seperti *def draw()*, *def animation(), def update()*.

Berikut adalah diagram dari *class Effect* :

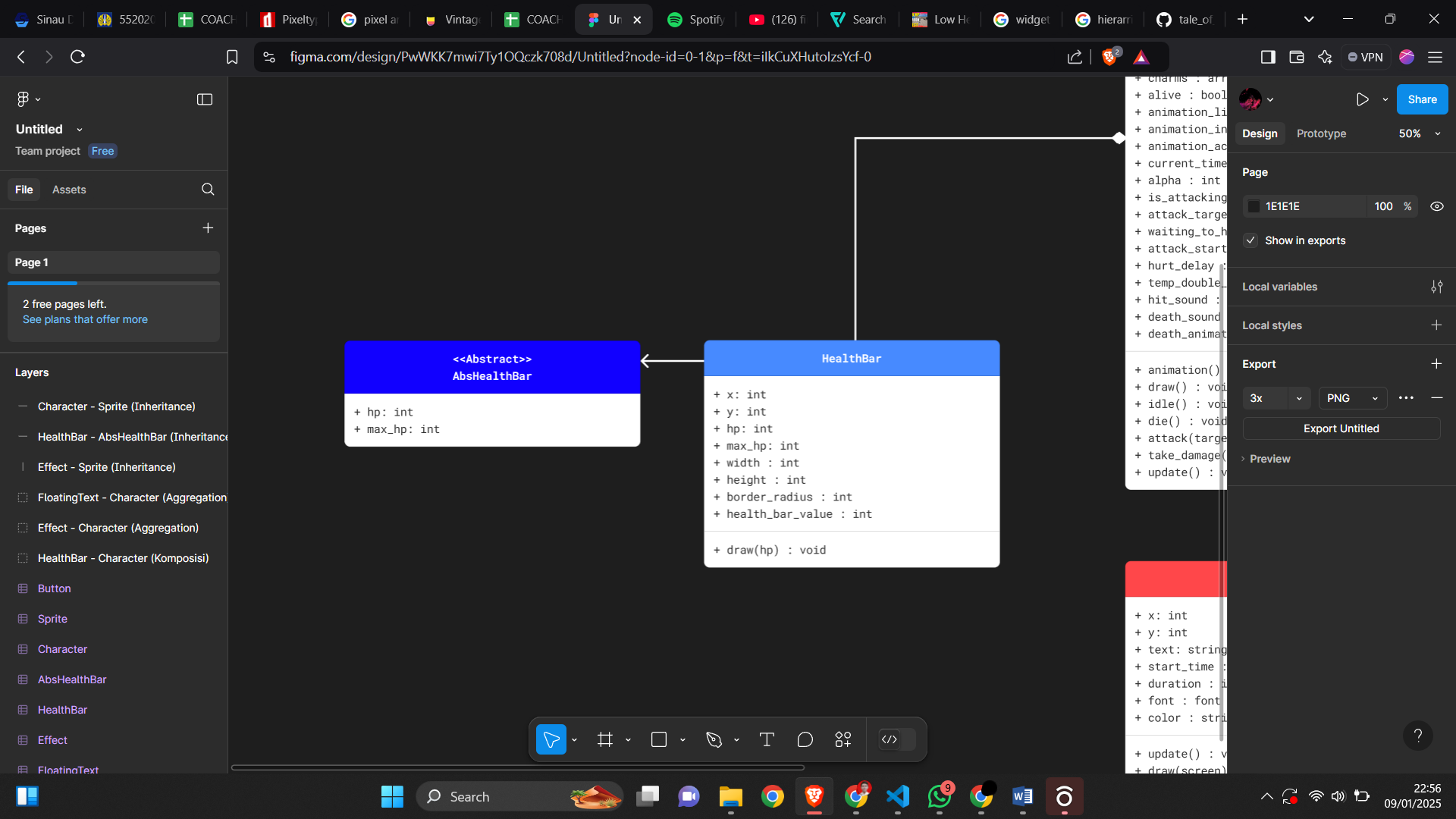


***Gambar 3.1.1.3*** *Diagram dari Class Effect (Sumber : Dokumentasi Penulis)*

#### *HealthBar* dan *AbsHealthBar*

*Health Bar* dibutuhkan untuk menampilkan *hp* yang tersisa dan dimiliki dari suatu *character*. Maka, *class HealthBar* merupakan komposisi dari *class character* karena keberadaannya tidak dapat dipisahkan dari *character*. Untuk mempermudah pembuatan *class HealthBar*, kami terlebih dahulu membuat *abstract class AbsHealthBar* yang terdiri atas *name, hp*, dan *max\_hp*.

Untuk menampilkan *healthbar* dalam bentuk *GUI*, maka kami menambahkan beberapa variabel seperti *x* dan *y* untuk menetapkan posisi dari *healthbar* yang akan dibuat, *width* untuk menentukan lebar *healthbar, height* untuk menentukan tinggi *healthbar, border\_radius* untuk memberikan sudut melengkung pada *healthbar*, dan *health\_bar\_value* untuk menunjukkan nilai *hp* yang tersisa. Serta fungsi *draw()* yang berfungsi untuk menggambar dan menampilkan *healthbar* secara visual. Berikut adalah diagram dari *class HealthBar* dan *class AbsHealthBar* :



***Gambar 3.1.1.4*** *Diagram dari Class HealthBar dan AbsHealthBar* *(Sumber : Dokumentasi Penulis)*

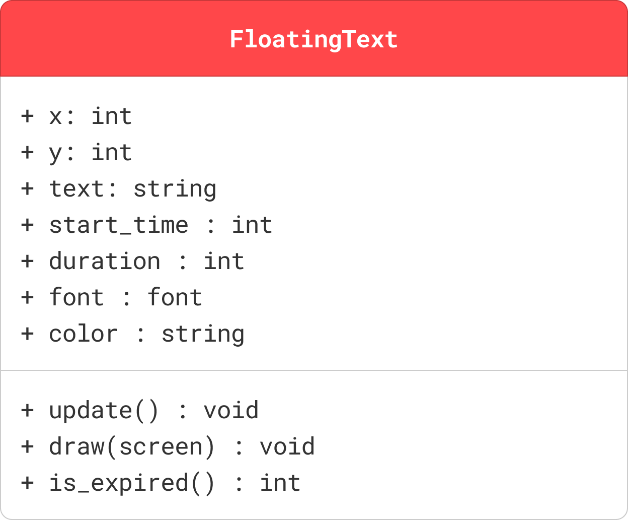
#### *FloatingText*

*Floating Text* adalah teks yang muncul secara mengambang yang memberitahukan informasi mengenai efek atau kejadian yang terjadi pada suatu *game*. Pada *game* kami, *floating text* digunakan untuk memberitahukan efek *charm* yang digunakan oleh *MC*. Sehingga, pemain mengetahui bahwa efek *charm* telah aktif.

Kami membuat *class FloatingText* agar menghindari penulisan kode berulang. Variabel yang digunakan terdiri atas *x* dan *y* untuk menetapkan posisi teks, *text* untuk menyimpan kata atau kalimat yang akan ditampilkan, *start\_time* untuk menyimpan data kapan teks ditampilkan, *duration* untuk menyimpan data seberapa lama teks ditampilkan, *font* untuk menggunakkan jenis font yang ditentukan, dan *color* untuk menentukan warna *font*.

*Class FloatingText* juga memiliki fungsi *def update()* untuk memperbarui keadaan teks, *def draw()* untuk menampilkan teks pada layar, *def is\_expired()* untuk menentukan kapan teks menghilang.

Berikut adalah diagram dari *class FloatingText* :

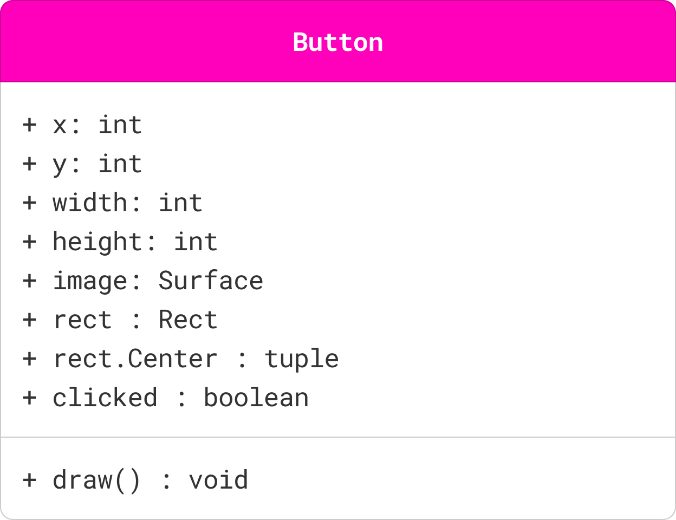
**

***Gambar 3.1.1.5*** *Diagram dari Class FloatingText (Sumber : Dokumentasi Penulis)*

#### *Button*

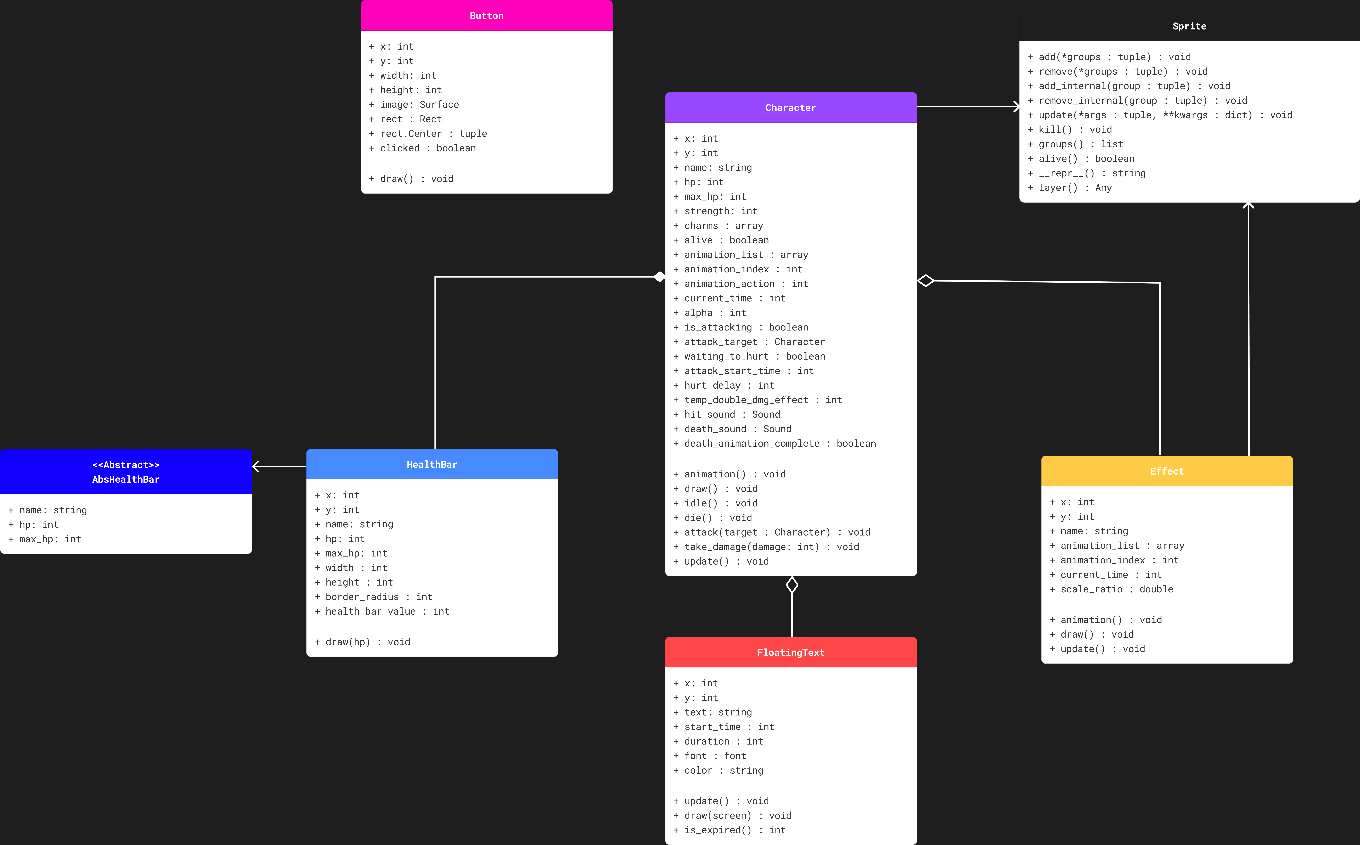
*Button* adalah elemen vital yang diperlukan dalam membuat aplikasi *GUI*. *Button* digunakan untuk mengakses berbagai fungsi yang ada pada program. Sehingga dibuatlah *class Button* memudahkan pengaksesan. *Class Button* memiliki variabel *x* dan *y* untuk menetapkan posisi *button*, *width* untuk menentukan lebar *button, height* untuk menentukan tinggi *button, image* untuk mengambil gambar dari *button*, *rect* untuk mendefinisikan dimensi dan posisi *button*, *clicked* untuk menyimpan status apakah *button* sudah diklik atau tidak.

Berikut adalah diagram dari *class Button*:



***Gambar 3.1.1.6*** *Diagram dari Class Button (Sumber : Dokumentasi Penulis)*

Dari semua *class* yang telah dirancang, dibuatlah *UML* (*Unified Modelling Language*) untuk memberikan gambaran mengenai struktur dan hubungan antar *class* dalam *game* kami. Berikut adalah UML dari *Tale of Samurai* :



***Gambar 3.1.1.7*** *UML dari Tale of Samurai (Sumber : Dokumentasi Penulis)*

### 3.1.2 *Library* dan Asetyang Digunakan

*Library* yang kami gunakan terdiri atas *PyGame*, *random,* dan *sys*. *PyGame* berfungsi untuk mempermudah proses perkembangan *game*. *Random*, dengan *func* *randint()*,  digunakan untuk mengambil angka acak yang nantinya akan dijumlahkan dengan stats *strength* dari karakter. *Sys*, dengan *func exit()*, digunakan untuk memberhentikan jalannya program setelah program selesai.

Kami menggunakkan cukup banyak aset yang dibagi ke dalam beberapa folder untuk menjaga keteraturan dan kemudahan pengelolaan. Kami melakukan pencarian dari banyak sumber dan memastikan bahwa aset-aset tersebut bebas hak cipta (*copyright free*) dan dapat digunakan tanpa perlu membayar. Aset tersebut terdiri atas :

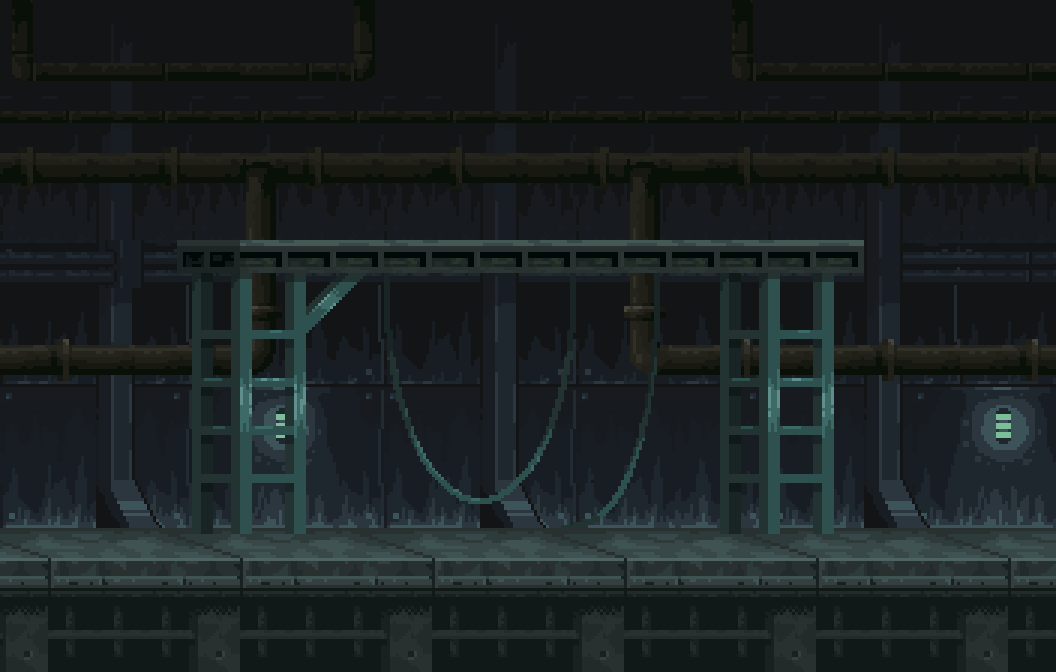
#### *Background*

Kami menggunakkan dua *background* yang akan digunakan sebagai latar belakang pada *game* kami. *mountain\_bg.png* digunakan sebagai *background* dari menu utama yang memberikan efek epik dan atmosferik untuk menyambut pemain. *mountain\_bg.png* kami temukan pada *craftpix.net* dengan judul *Free Mountain Backgrounds Pixel Art*.



***Gambar 3.1.2.1*** *mountain\_bg.png sebagai background dari menu utama (Sumber : https://craftpix.net/freebies/free-mountain-backgrounds-pixel-art/)*

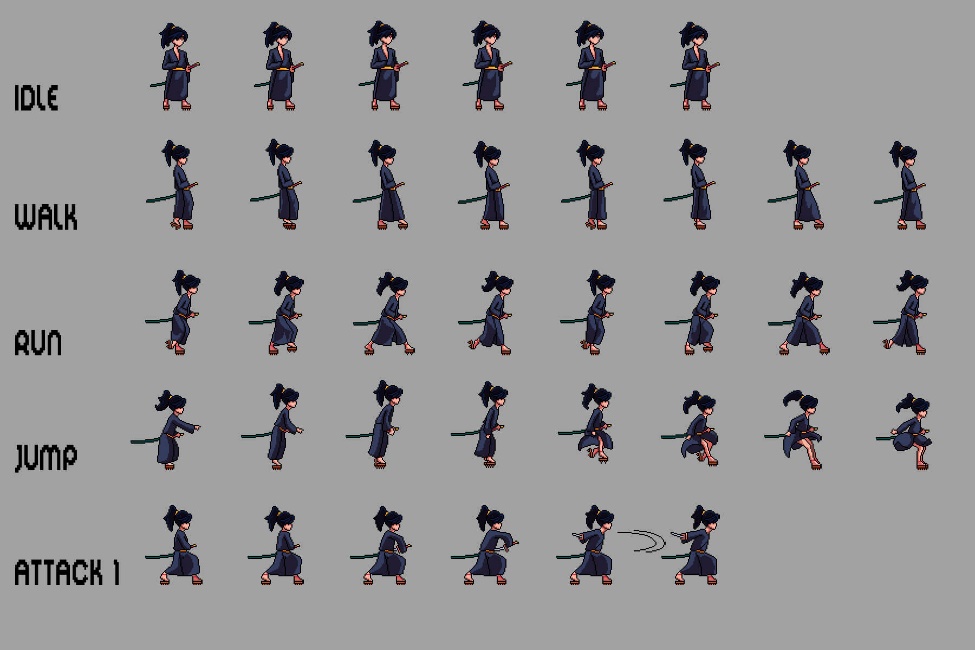
*background.png* digunakan sebagai *background* dari *fight scene*, nuansa *background.png* dipilih gelap agar memberi suasana yang menegangkan karena samurai melawan *yokai. background.png* kami temukan pada *pinterest* dengan judul *Bulkhead Walls Hangar* karya Bradley Wascher.



***Gambar 3.1.2.2*** *background.png sebagai background dari fight scene (Sumber : https://www.pinterest.com/pin/bulkhead-walls-hangar--530228556121612535/)*

#### *Character*

Sesuai pembahasan pada 3.1, kami menggunakkan samurai dan *yokai* sebagai karakter pada *game* kami. Kami mencari aset *character* dari website *craftpix.net* Untuk karakter samurai, kami memilih desain sederhana–samurai tanpa jubah yang hanya dilengkapi dengan pedang. Aset yang kami temukan untuk karakter samurai memiliki banyak *folder* yang terdiri atas *folder frame* animasi *attack\_1, attack\_2, attack\_3 dead, hurt, idle, jump, run, walk*. Agar efisien, kami hanya menggunakkan  *folder frame idle, attack\_1, dead*.



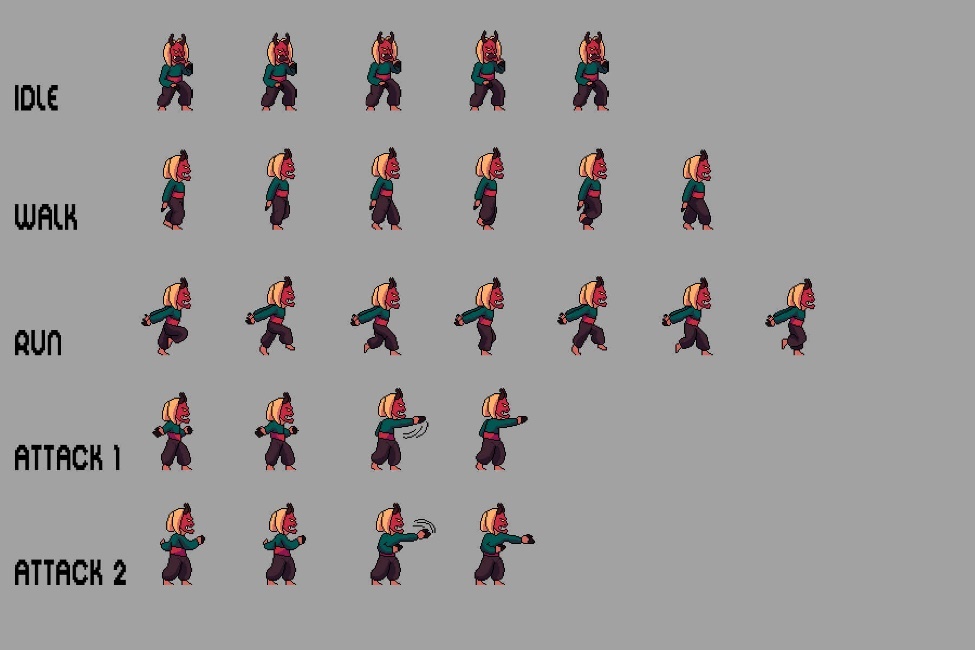
***Gambar 3.1.2.3*** *frame* animasi dari karakter Samurai *(Sumber : https://craftpix.net/freebies/free-shinobi-sprites-pixel-art/)*

Dari berbagai jenis *yokai* ada, kami memilih *yorei* dan *gotoku* sebagai musuh yang dihadapi oleh *MC*. *Yorei* adalah hantu perempuan bermuka pucat yang berpakaian serba putih, memiliki rambut panjang dan hitam. Aset yang kami temukan untuk karakter samurai memiliki banyak *folder* yang terdiri atas *folder frame* animasi *attack\_1, attack\_2, attack\_3 dead, hurt, idle, jump, run, walk*. Agar efisien, kami hanya menggunakkan *folder frame idle, attack\_1, dead*.



***Gambar 3.1.2.4*** *frame* animasi dari karakter Samurai *(Sumber : https://craftpix.net/freebies/free-shinobi-sprites-pixel-art/)*

*Gotoku* adalah hantu yang menggunakkan topeng oni. Oni adalah makhluk mitologi Jepang yang melambangkan kekuatan dan kejahatan. Dengan topeng oni, Gotoku terlihat lebih menakutkan dan menyeramkan.  Aset yang kami temukan untuk karakter samurai memiliki banyak *folder* yang terdiri atas *folder frame* animasi *attack\_1, attack\_2, attack\_3 dead, hurt, idle, jump, run, walk*. Agar efisien, kami hanya menggunakkan *folder frame idle, attack\_1, dead*.



***Gambar 3.1.2.5*** *frame* animasi dari karakter *Gotoku (Sumber : https://craftpix.net/freebies/free-shinobi-sprites-pixel-art/)*

#### *Buttons*

Aplikasi *GUI* tentunya membutuhkan *button* untuk mempermudah pengaksesan. Kami mencari aset *button* dari *website itch.io* dan menemukan aset yang sesuai. Oleh karena itu, kami menggunakan aset *button btn\_play,* *btn\_exit*, *btn\_menu,, btn\_retry.*

Pada halaman utama kami, *btn\_play.png* bisa digunakan oleh pemain ketika ingin memulai *game*.

https://lh7-rt.googleusercontent.com/docsz/AD_4nXcnwf9bhPq81qvnSgYAZbQ8q2Z8ea8XIVg19OLP3_fyE1SktzW3eYAOQ-rFTrh8F82IJxFspM9GozNgqaBu__1O9AnsUoM-vcWY8SwbyrQ_hr0q8MD0P6W5KD4zUtex18ReOQm8GA?key=LnUhC1axidrVlH4Ro7NhuWnH

***Gambar 3.1.2.6*** *btn\_play.png untuk memulai game (Sumber : https://free-game-assets.itch.io/tds-pixel-art-2d-kit)*

Pada halaman utama kami, di sebelah kiri atas terdapat *button* keluar *btn\_exit.png* bisa digunakan oleh pemain ketika ingin keluar dari *game*.

https://lh7-rt.googleusercontent.com/docsz/AD_4nXdyOUvgxlT-ocN1IKBOPcor3aWVyBbZSt7P1A9Zm3cNB_pVwwWm85OSrtleCSyRJG16HxPgllo3kmPWxTxJHwUenMIO4kcWwA22FGiva7pEJHqO1deL3E1Kqbca0QE8a2X9Eut6Ag?key=LnUhC1axidrVlH4Ro7NhuWnH

***Gambar 3.1.2.7*** *btn\_exit.png untuk mengakhiri game (Sumber : https://free-game-assets.itch.io/tds-pixel-art-2d-kit)*

Ketika *game* selesai, maka akan muncul *button* menu *btn\_menu.png* yang bisa digunakan oleh pemain ketika ingin kembali ke menu awal.

https://lh7-rt.googleusercontent.com/docsz/AD_4nXc-3hNAwlvGBmDCR7-8IKaVx0yvS5Rnt7S6qIUp5bp9bYVY3akX79PjHXq9RwgE2McbreeqHO0oL9J3OrMaVJNeFz4CiIc--B_NkKROJDs8SJuhKlTPQwj4BUVCmUnds9jLLr1R?key=LnUhC1axidrVlH4Ro7NhuWnH

***Gambar 3.1.2.8*** *btn\_menu.png untuk mengakses menu utama(Sumber : https://free-game-assets.itch.io/tds-pixel-art-2d-kit)*

Di atas *btn\_menu.png*, maka akan muncul *button* *retry btn\_retry.png* yang bisa digunakan oleh pemain ketika ingin mengulang *game*.

https://lh7-rt.googleusercontent.com/docsz/AD_4nXegdQP7gIgVk3Mb4JV6ru9Wa5Qa25Xpr8GFJvYvSxrc8dix3TfT_DfYKHs73JUMLRMAUiVuCdUnDN2cb2dSFIlzHLvU4bTEs8NCgcWfcrPSkmVyKvLTJgr0cinJDCNvYLiF_k7k?key=LnUhC1axidrVlH4Ro7NhuWnH

***Gambar 3.1.2.9*** *btn\_retry.png untuk mengulang game (Sumber : https://free-game-assets.itch.io/tds-pixel-art-2d-kit)*

#### *Charms*

Dalam *game* kami, Samurai memiliki *charm* yang digunakan sebagai kekuatan tambahan ketika bermain. Kami melakukan pencarian aset *charms* pada *website flaticon.com*. Aset yang digunakan berupa gambar *pixelated* yang sesuai dengan kegunaannya masing-masing. Kotak kosong di bagian kanan bawah digunakan untuk menampilkan jumlah *charm* tersisa yang bisa digunakan.

*Flask.png* digunakan sebagai aset *charm* untuk Samurai dapat mengisi *hp* miliknya (*refill\_heart*). Pemilihan ini didasarkan pada representasi umum dalam banyak *game* yang menggunakkan *flask* sebagai simbol untuk menambah HP. Maka diharapkan pemain dapat mengetahui bahwa *flask.png* merupakan *charm* untuk menambah *hp MC.*



***Gambar 3.1.2.10*** *Flask.png untuk refill\_heart (Sumber : https://www.flaticon.com/free-icon/flask\_12751685?term=pixel+art&page=1&position=54&origin=search&related\_id=12751685)*

*Double\_sword.png* digunakan sebagai aset *charm* yang memungkinkan Samurai untuk menggadakan nilai *attack* sebanyak dua kali. Pemilihan ini didasarkan pada representasi umum dalam banyak *game* yang menggunakkan pedangsebagai simbol untuk menambah *damage*. Maka diharapkan pemain dapat mengetahui bahwa *double\_sword.png* merupakan *charm* untuk menambah serangan *MC*.



***Gambar 3.1.2.11*** *Double\_sword.png untuk double\_damage (Sumber : https://www.flaticon.com/free-icon/game\_13929630?term=sword+pixel+art&page=1&position=1&origin=search&related\_id=13929630)*

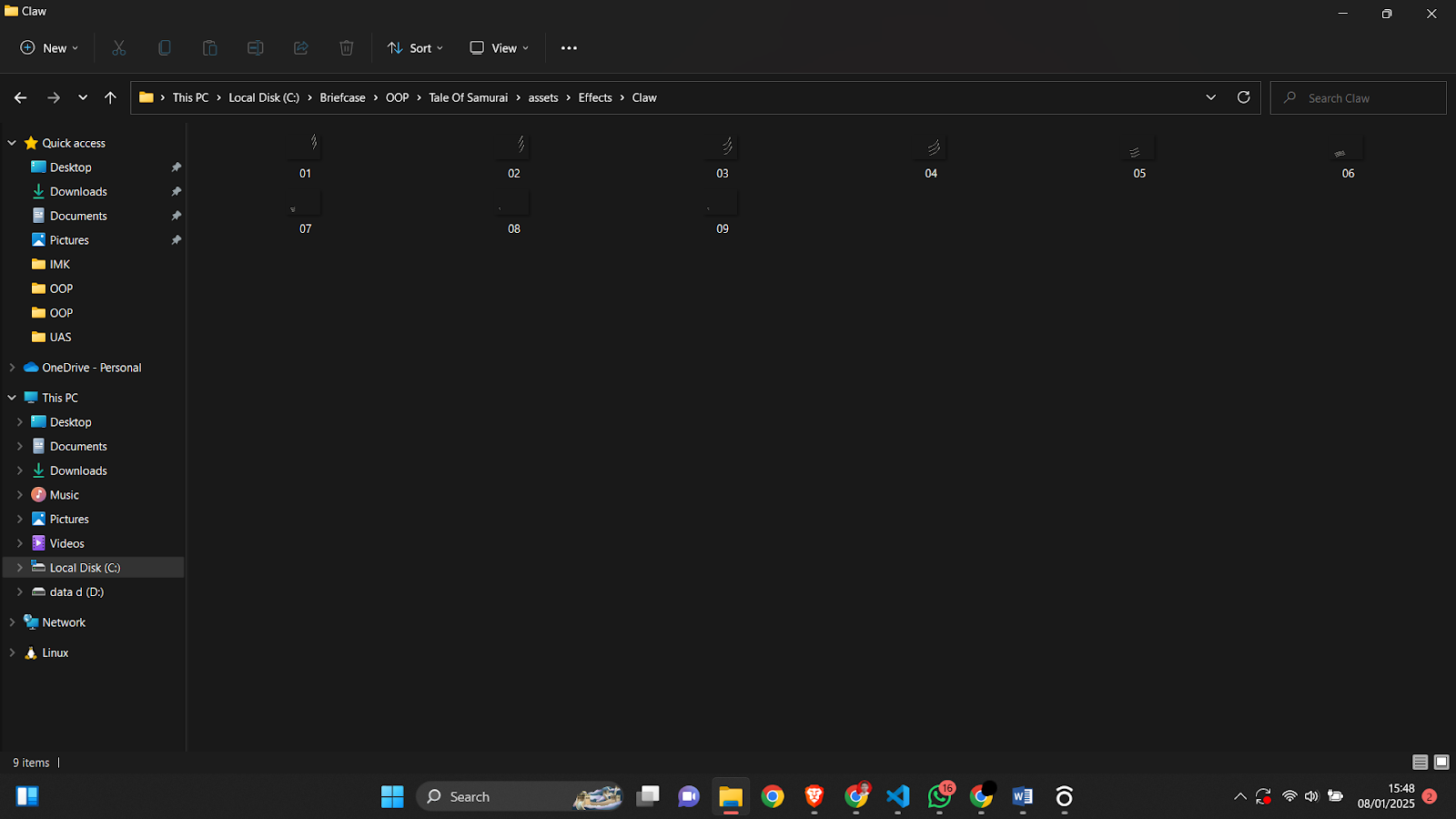
#### *Effects*

Sesuai pembahasan pada 3.1.1 pada *class Effects*, *character* memiliki serangan *effects*. Kami memutuskan agar setiap *character* memiliki *effect* yang sesuai dengan jenis serangannya. Samurai mendapat *effect* *slash* karena menggunakkan pedang. Untuk *yorei* diberikan *effect blood* karena menyerang dari jarak jauh dan *gotoku* diberikan *effect claw* karena menyerang menggunakkan cakar.

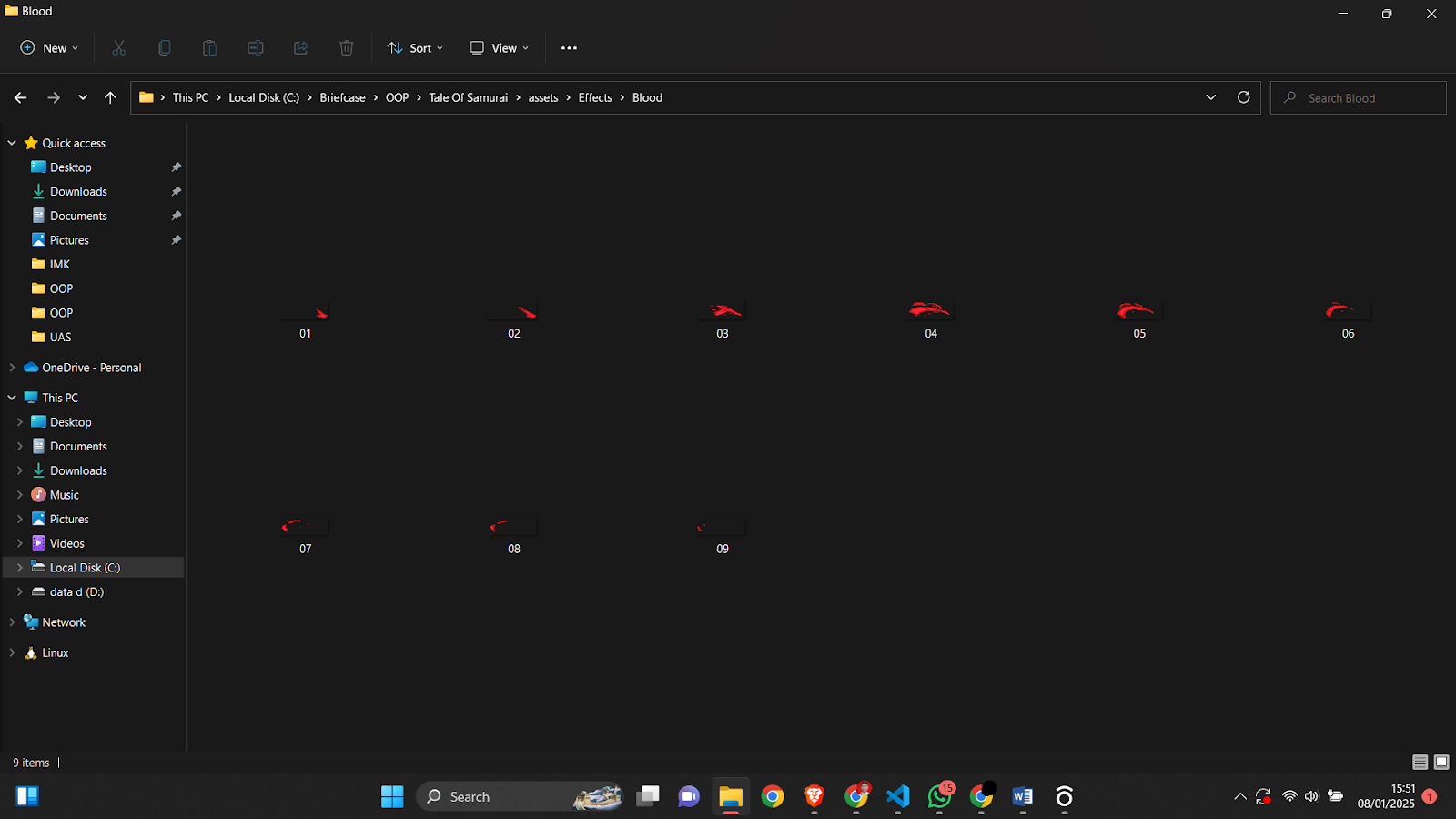
Kami menemukan *effects* yang sesuai pada *website itch.io* yang dibuatoleh NYKNCK. Berikut adalah efek-efek tersebut :



***Gambar 3.1.2.12*** *Slash effects* yang akan digunakan untuk samurai *(Sumber : https://nyknck.itch.io/slash)*



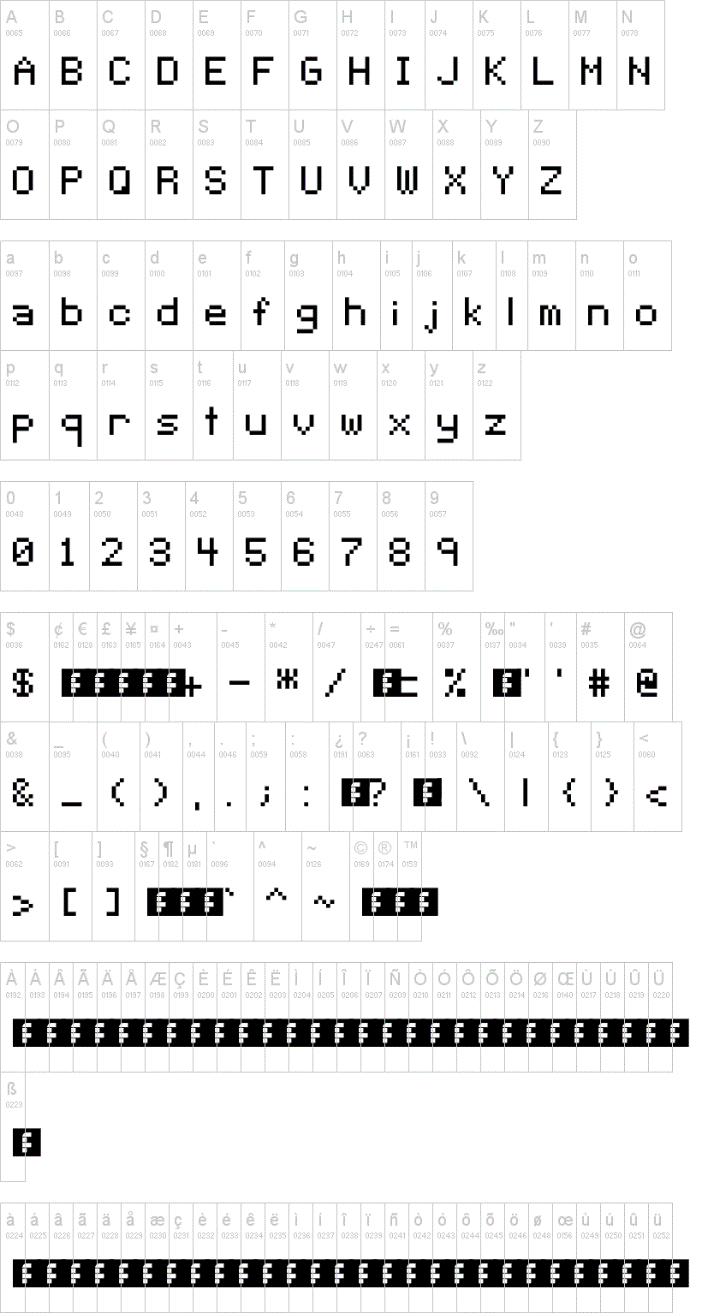
***Gambar 3.1.2.13*** *Claw effects* yang akan digunakan untuk *gotoku (Sumber : https://nyknck.itch.io/clawsct)*



***Gambar 3.1.2.14*** *Blood effects* yang akan digunakan oleh *yorei (Sumber : https://nyknck.itch.io/bloodfx)*

#### *Font*

Suatu aplikasi tentu memerlukan *font* yang nyaman dibaca dan sesuai dengan tema aplikasi. Oleh karena itu, kami memilih menggunakkan *Pixeltype.ttf* sebagai satu-satunya *font* yang dalam *game* kami. Alasannya karena *Pixeltype.ttf* adalah *font* yang sesuai dengan gaya grafik pada *game* kami, yaitu *pixelated*, sehingga menciptakan keselarasan visual antara teks dan aset lainnya.



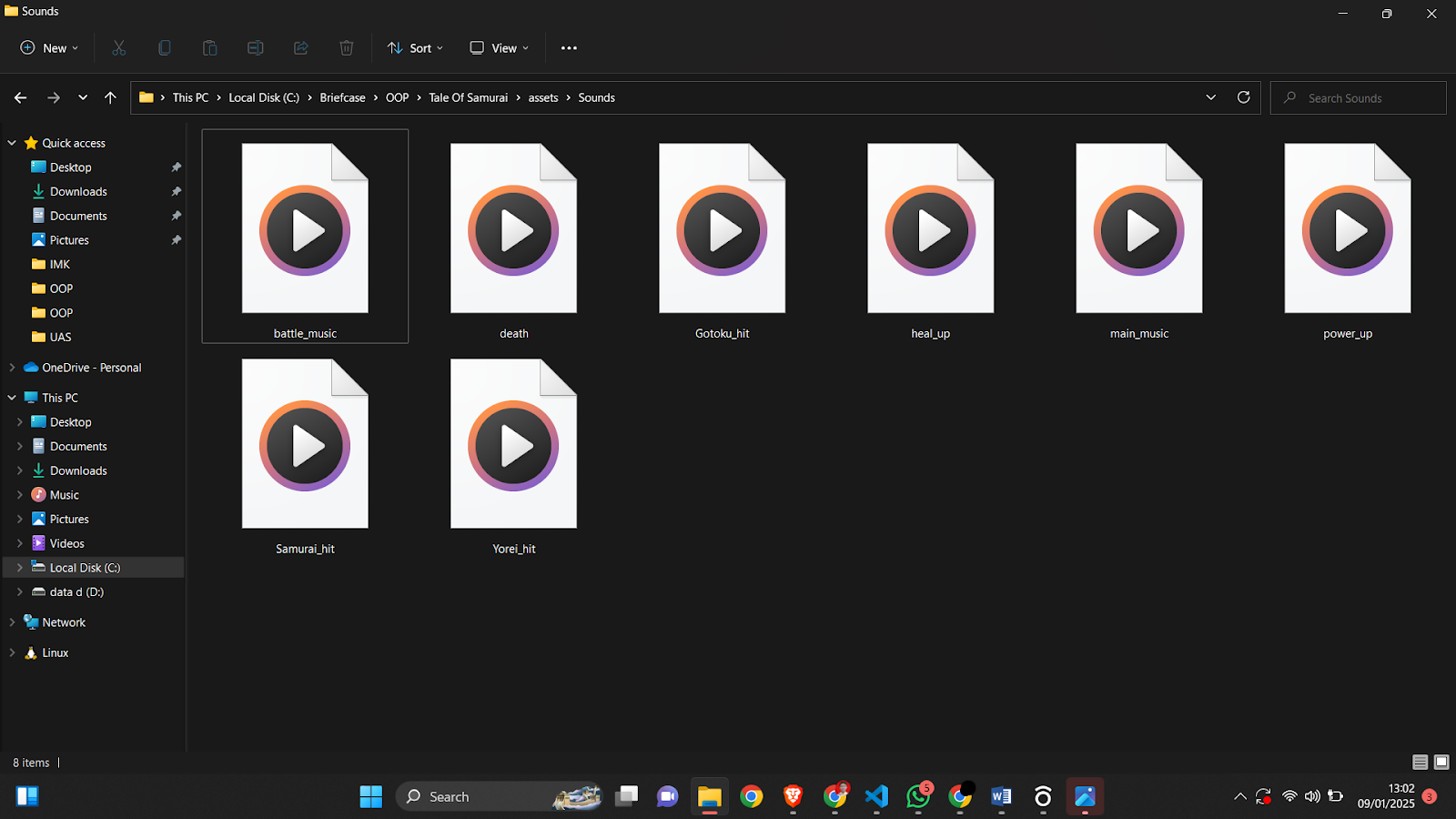
***Gambar 3.1.2.15*** *Tampilan font Pixeltype.ttf yang akan digunakan pada game kami (Sumber :* [*https://www.dafont.com/pixeltype.font*](https://www.dafont.com/pixeltype.font)*)*

#### *Sounds*

*Game* tentu memerlukan musik atau *sound effect* agar menciptakan suasana yang lebih hidup atau enerjik. Tidak jarang suatu musik atau *sound effect* yang tepatdapat membuat suatu *game* menjadi ikonik. Maka dari itu, kami melakukan beberapa pencarian pada *youtube* untuk mencari musik dan *sound effect* yang sesuai.

Pertama, kami mencari musik yang akan dimainkan pada *main menu* dan saat pertandingan berlangsung. Musik untuk *main menu* dipilih secara atmosferik dan tentunya sesuai dengan grafik yang dibawakan, yaitu *pixel type song 8 bit*. Kami menemukan lagu yang sesuai dari xDeviruchi dengan lagu “*Title Theme”* dan “*Prepare for Battle!”*

Sedangkan untuk *sound effect* seperti serangan karakter menggunakkan pedang, cakaran, penggunaan *charm*, suara karakter saat mati. Kami melakukan pencarian pada *pixabay* dan mencari *sound effect* yang sesuai. Sehingga, terdapat dua musik dan enam *sound effect* yang digunakan dan berikut adalah hasil pencarian kami :



***Gambar 3.1.2.16*** *Musik dan sound effect yang akan digunakan pada game kami (Sumber : Dokumentasi penulis)*

#### Aset lainnya

Terdapat aset tambahan lainnya yang tidak kami masukkan ke dalam *folder* terpisah karena hanya terdiri atas satu *file*, seperti *panel.png, sword.png, low\_health\_screen.png*. Semua aset tambahan ini tentunya menggunakkan gaya grafik *pixelated* agar tetap konsisten dan sesuai dengan tema *game* yang kami usung. *Panel.png* digunakan sebagai latar untuk *healthbar* dari *MC* dan musuh, serta *charm MC*.



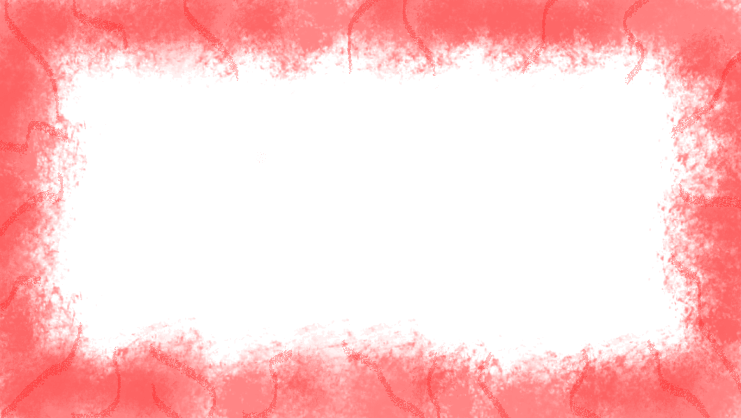
***Gambar 3.1.2.17*** *panel.png yang akan digunakan sebagai latar untuk healthbar MC dan musuh (Sumber : Dokumentasi penulis)*

*Sword.png* digunakan sebagai kursor pada *game*. Ikon kursor akan berubah menjadi *sword* apabila kursor diarahkan pada area musuh (*rect* musuh). Sehingga, pemain dapat mengetahui bahwa musuh dapat diserang saat giliran pemain untuk menyerang.

*https://lh7-rt.googleusercontent.com/docsz/AD_4nXfXHutmanHUit1WXVtAqQbvYttIegmjiOYyMZ_E8KmwRZWEEp8l1UN3Z7WM9IEEdX4PMvbtXNqeJIEhi6D5G0r8P_QYW2nQU8zl8S_7qQSIbe9EMkUEF-YkIa-QfQLwn5hSDTwaUQ?key=LnUhC1axidrVlH4Ro7NhuWnH*

***Gambar 3.1.2.18*** *sword.png yang akan digunakan sebagai kursor pada game (Sumber : https://www.flaticon.com/search?word=sword%20pixel)*

*Low\_health\_screen.png* digunakan untuk memberikan indikasi apabila *MC* memiliki *hp* dibawah seperempat dari *max\_hp*. Tampilan ini digunakan untuk memperingatkan pemain bahwa *hp* *MC* sedang dalam kondisi kritis, sehingga pemain dapat lebih waspada untuk mengambil langkah selanjutnya.



***Gambar 3.1.2.19****low\_health\_screen.png yang akan muncul apabila hp MC sedang dalam kondisi kritis (Sumber : https://that-skye.itch.io/low-health-on-screen-effect)*

Terdapat aset tambahan lainnya yang kami buat sendiri menggunakkan *photoshop*, yaitu *game\_title.png*, *victory.png* dan *defeat.png*. Untuk aset tambahan tersebut, kami menggunakkan gaya grafik *pixelated* agar sesuai dengan tema *game* kami. Untuk *title.png,* sesuai dengan judul *game* yang telah disepakati sebelumnya, *Tale of Samurai* dibuat dengan efek *pixelated 3D,* dengan tambahan pedang katana di bagian tengah. *Title.png* juga diberikan gradasi warna putih-merah dan putih-biru. *Title.png* juga digunakan sebagai *icon widget* ketika aplikasi dijalankan, menggantikan logo *pygame* yang berupa ular berwarna kuning.



***Gambar 3.1.2.20*** *game\_title.png yang akan menjadi title dari game “Tale of Samurai” (Sumber : Dokumentasi Penulis)*

Sama seperti *title.png*, *victory.png* dibuat dengan efek *pixelated 3D.* *Victory.png* juga diberikan gradasi warna putih-kuning cerah yang memberi efek eforia dan pencapaian kepada pemain, seolah menegaskan bahwa pemain telah memenangkan permainan.



***Gambar 3.1.2.21*** *victory.png yang muncul ketika pemain memenangkan game (Sumber : Dokumentasi Penulis)*

Untuk *defeat.png*, *defeat.png* dibuat dengan efek *pixelated 3D* dan diberikan efek gradasi warna hitam-biru gelap yang memberi efek kelam, mempertegas bahwa pemain telah kalah. Untuk mendramatisir, terdapat siluet katana yang patah yang seolah-olah melambangkan bahwa *MC* telah gugur dalam perjuangnnya.



***Gambar 3.1.2.22*** *defeat.png yang muncul ketika pemain kalah (Sumber : Dokumentasi Penulis)*

## 3.2 Program Utama

Disini kami akan akan menjelaskan mengenai program utama yang ada dibelakang game yang kami buat ini. Program utama ini merupakan blok kode perulangan(*looping*) yang berada di dalam fungsi *play()*. Berikut blok kode program nya:

|  |
| --- |
| while running:          for event in pygame.event.get():              if event.type == pygame.QUIT:                  running = False                  pygame.quit()                  exit()              if game\_active and event.type == pygame.MOUSEBUTTONDOWN:                  clicked = True              else:                  clicked = False |

Pada blok kode perulangan diatas, kami melakukan pengecekan *event* yang ada dalam *library pygame* yang mana terdapat *event QUIT* yang berfungsi untuk stop program dan keluar dari *state* permainan yang sedang berlangsung. Disamping itu, *MOUSEBUTTONDOWN* seperti namanya, ini menerima input dari pengguna yang berupa *click mouse*.

|  |
| --- |
| mouse\_pos = pygame.mouse.get\_pos()          for enemy in enemies:              if current\_character == 1:                  if enemy.alive and enemy.rect.collidepoint(mouse\_pos):                      pygame.mouse.set\_visible(False)                      screen.blit(sword\_surf, mouse\_pos)                  else:                      pygame.mouse.set\_visible(True)              else:                  pygame.mouse.set\_visible(True) |

Blok kode ini berfungsi untuk memberikan perubahan tampilan *cursor* pada *mouse* menjadi ikon pedang apabila bertindihan/bertabrakan dengan gambar/objek musuh.

|  |
| --- |
| if current\_character > total\_characters:              current\_character = 1              total\_characters = 1 + sum(1 for enemy in enemies if enemy.alive) |

Lalu, untuk memastikan *variable current\_character* tidak melewati batas dari jumlah *total\_characters*(yang mana dalam kasus game ini berjumlah 3 karakter) dan memberikan *value* kepada *variable total\_characters* yang akan terus berubah apabila salah satu musuh terkalahkan.

|  |
| --- |
| attack = False          target = None          for count, enemy in enumerate(enemies):              if enemy.rect.collidepoint(mouse\_pos):                  if clicked:                      attack = True                      target = enemies[count] |

Blok kode diatas memberikan *value* kepada *variable target* yang akan ditentukan berdasarkan pilihan *player* terhadap musuh mana yang akan diserang(ditandai dengan *collide point* dari *cursor mouse*).

|  |
| --- |
| heath\_charm\_btn = button.Button(100, (screen\_height + bottom\_panel\_height / 2) + 30, health\_charm\_img, 0.09)          doubledmg\_charm\_btn = button.Button(100 + 50, (screen\_height + bottom\_panel\_height / 2) + 30, doubledmg\_charm\_img, 0.09)            heal\_confirm = False          double\_damage\_confirm = False          if heath\_charm\_btn.draw(screen):              samurai.charms["refill\_health"]["active"] = True              heal\_confirm = True          elif doubledmg\_charm\_btn.draw(screen):              double\_damage\_confirm = True              samurai.charms["double\_damage"]["active"] = True |

Blok kode diatas berfungsi untuk membuat kedua tombol untuk menampilkan sekaligus mengaktifkan fungsi *charms* atau kemampuan tambahan untuk *player*. Fungsi *nama\_tombol.draw(screen)* mengembalikan nilai *boolean*, sehingga apabila bernilai *true*, seorang *player* terindikasi telah mengaktifkan salah satu *charms* dan memberikan nilai *true* pula kepada *variable* *health\_confirm/double\_damage\_confirm*.

|  |
| --- |
| if samurai.alive:              if current\_character == 1:                  action\_delay = 50                  action\_cooldown += 1                  if action\_cooldown >= action\_delay:                      if attack and target is not None:                          samurai.attack(target)                          active\_effects.append(Effect(target.rect.centerx, target.rect.centery, "Slash"))                          if samurai.temp\_double\_dmg\_effect > 1:                              samurai.temp\_double\_dmg\_effect = 1                          current\_character += 1                          action\_cooldown = 0                        # healing charm                      if samurai.charms["refill\_health"]["active"]:                          if samurai.charms["refill\_health"]["amount"] > 0 and heal\_confirm:                              if samurai.max\_hp - samurai.hp > samurai.charms["refill\_health"]["effect"]:                                  heal\_amount = samurai.charms["refill\_health"]["effect"]                              else:                                  heal\_amount = samurai.max\_hp - samurai.hp                              samurai.hp += heal\_amount                              heal\_up\_sound.play()                              floating\_texts.append(FloatingText(                                  f"+{heal\_amount} HP",                                  samurai.rect.centerx,                                  samurai.rect.top - 20,                                  3000,                                  fonts['sm'],                                  colors['white']                              ))                              samurai.charms["refill\_health"]["amount"] -= 1                              current\_character += 1                              action\_cooldown = 0                              samurai.charms["refill\_health"]["active"] = False                              heal\_confirm = False                        # double damage charm                      if samurai.charms["double\_damage"]["active"]:                          if samurai.charms["double\_damage"]["amount"] > 0 and double\_damage\_confirm:                              samurai.temp\_double\_dmg\_effect = samurai.charms["double\_damage"]["effect"]                              double\_damage\_sound.play()                              floating\_texts.append(FloatingText(                                  f"x{samurai.charms['double\_damage']['effect']} Damage",                                  samurai.rect.centerx,                                  samurai.rect.top - 20,                                  3000,                                  fonts['sm'],                                  colors['white']                              ))                              samurai.charms["double\_damage"]["amount"] -= 1                              current\_character += 1                              action\_cooldown = 0                              samurai.charms["double\_damage"]["active"] = False                              double\_damage\_confirm = False |

Lalu, berikut merupakan blok kode terpanjang dan terkompleks yang ada pada seluruh program yang kami buat. Blok kode ini merupakan rentetan logika dari *player*. Dimulai dari indikasi apabila *player* masih hidup, maka game akan terus berjalan dan memberikan *player* hak untuk melakukan serangan ataupun memakai *charms* yang masih tersedia.

Disini, setiap kali *player* melakukan sebuah serangan, ia akan diberikan waktu *cooldown* yang bertujuan untuk memberikan efek *delay* pada animasi serangan.

|  |
| --- |
| action\_delay = 100          for count, enemy in enumerate(enemies):              if current\_character == 2 + count:                  if enemy.alive:                      action\_cooldown += 1                      if action\_cooldown >= action\_delay:                          enemy.attack(samurai)                          if enemy.name == "Yorei":                              active\_effects.append(Effect(samurai.rect.centerx, samurai.rect.centery, "Blood"))                          elif enemy.name == "Gotoku":                              active\_effects.append(Effect(samurai.rect.centerx, samurai.rect.centery, "Claw"))                          current\_character += 1                          action\_cooldown = 0                  else:                      current\_character += 1 |

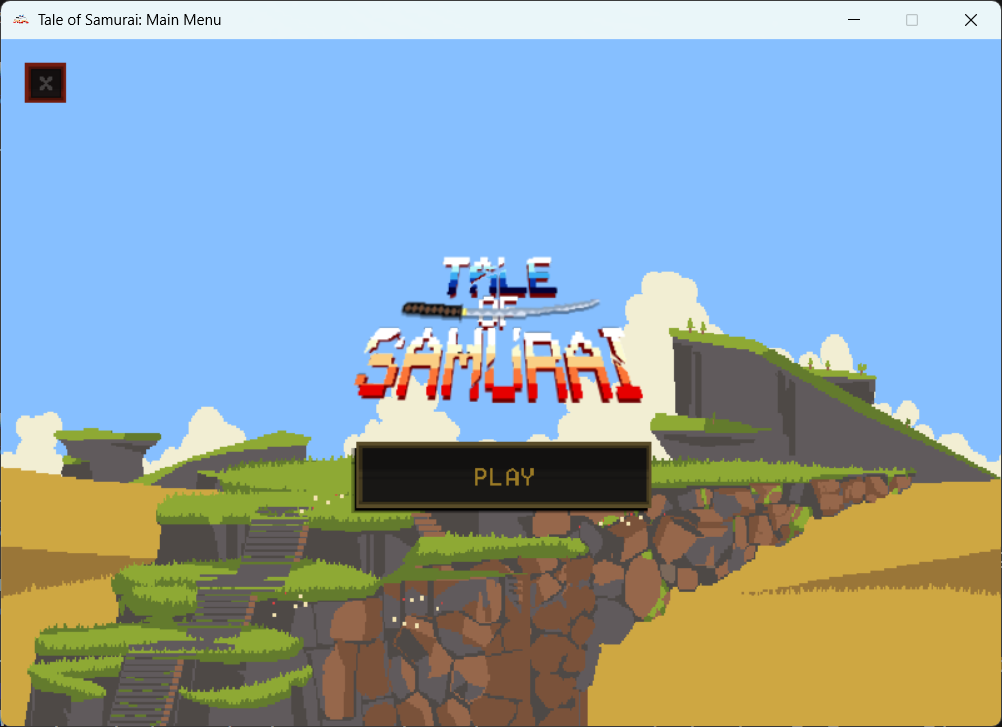
Sama seperti blok kode sebelumnya yang berfungsi untuk memberikan logika serangan maupun penggunaan *charms* pada *player*. Kali ini, logika yang sama diterapkan kepada musuh-musuh(*computer*) yang ada. Namun, dikarenakan musuh-musuh yang ada game ini tidak memiliki kemampuan untuk mennggunakan *charms*, maka logika penggunaan *charms* tidak kami implementasikan kedalam blok kode ini.

|  |
| --- |
| if samurai\_health\_bar.hp <= 0:              game\_state = "game\_over"              restart(game\_state)          elif gotoku\_health\_bar.hp <= 0 and yorei\_health\_bar.hp <= 0:              game\_state = "victory"              restart(game\_state) |

Lalu, blok kode terakhir namun juga tidak kalah penting ialah penentu hasil *battle*. Seperti dapat dilihat pada blok kode tersebut, apabila status *health bar* dari samurai(*player*) kurang dari 0, maka hasil *battle* adalah *game\_over* dan sebaliknya. Kedua kondisipun tetap mengeksekusi fungsi *restart()* yang sekaligus mengarahkan *player* ke menu *restart*.

## 3.3 Dokumentasi Program

### 3.3.1 Halaman awal permainan



***Gambar 3.3.1.1*** *Menu utama dari Tale of Samurai* *(Sumber : Dokumentasi Penulis)*

Halaman ini merupakan tampilan awal ketika membuka aplikasi ini untuk pertama kali. Terdapat *Gambar 3.1.2.1* sebagai aset yang digunakan sebagai latar belakang. Di bagian kiri atas terdapat tombol yang bisa digunakan ketika ingin keluar dari permainan. Di bagian tengah halaman, terdapat judul permainannya yaitu *Tale of Samurai* serta tombol *play* di bawahnya yang bisa digunakan ketika ingin memulai permainan.

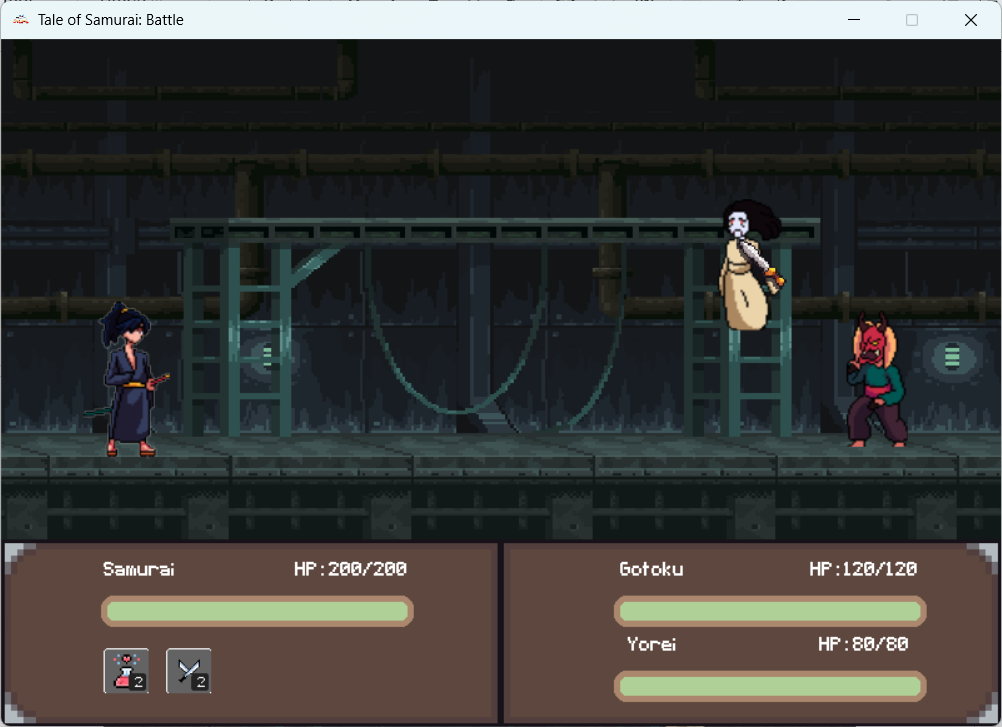
### 3.3.2 Tampilan Program



***Gambar 3.3.1.2*** *Halaman Permainan Awal dari Tale of Samurai* *(Sumber : Dokumentasi Penulis)*

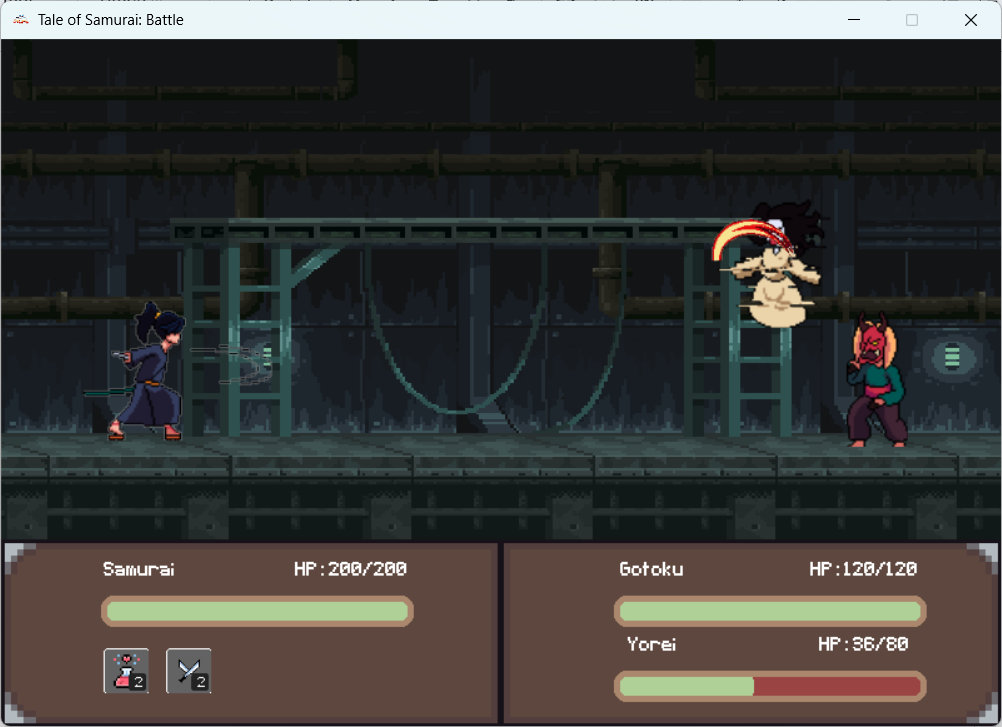
Ini merupakan tampilan ketika memulai permainan. Di sisi kiri terdapat karakter samurai sebagai karakter utama dalam permainan ini dan itu juga merupakan karakter yang akan dimainkan oleh pengguna. Di sisi kanan terdapat dua musuh yaitu Yorei, karakter yang terbang dan juga Gotoku, karakter yang mengenakan topeng berwarna merah.

Di bagian bawah terdapat juga nama karakter yang ada pada permainan serta jumlah HP mereka. Untuk karakter samurai sendiri diperlengkapi dengan *charm*, semacam kekuatan tambahan yang berada di bawah HP *bar*. Samurai memiliki dua *charm*, yaitu *charm* untuk melakukan pengisian HP dan juga charm untuk mengalikan *damage* serangan sebanyak 2.



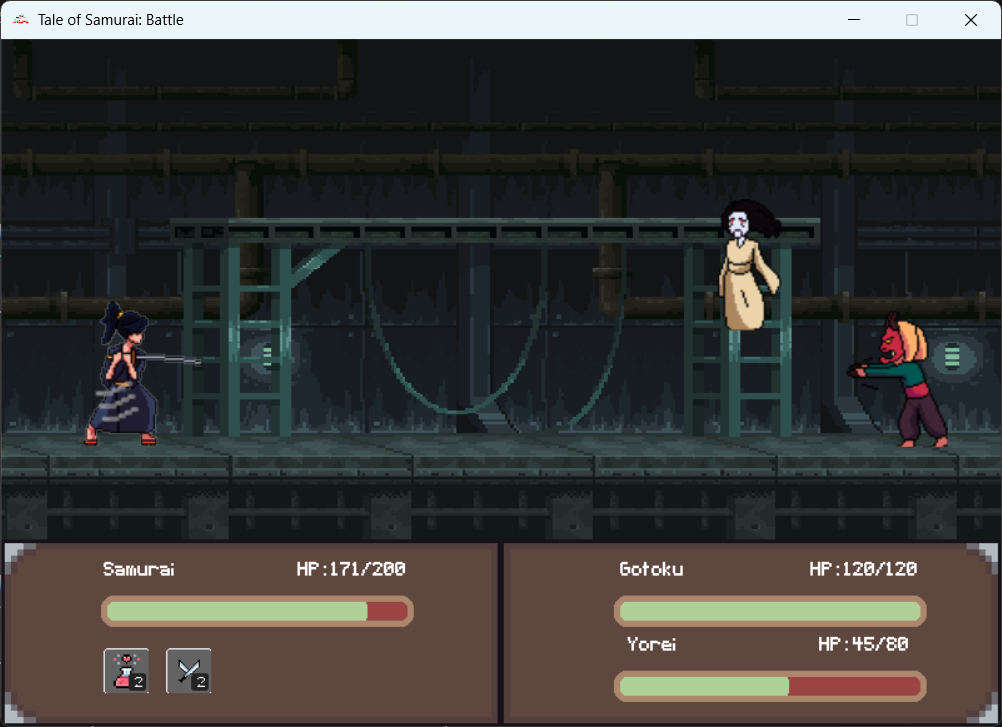
***Gambar 3.3.1.3*** *Animasi kursor menjadi Pedang* *(Sumber : Dokumentasi Penulis)*

Ketika memulai permainan, Samurai akan diberi kesempatan terlebih dahulu untuk melakukan serangan. Pemain bisa mengarahkan kursorke karakter lawan yang ingin diserang, ketika diarahkan, maka tampilan kursorakan berubah menjadi pedang yang menandakan bahwa pemain bisa melakukan serangan. Pemain hanya perlu melakukan klik pada *mouse* untuk melakukan serangan.



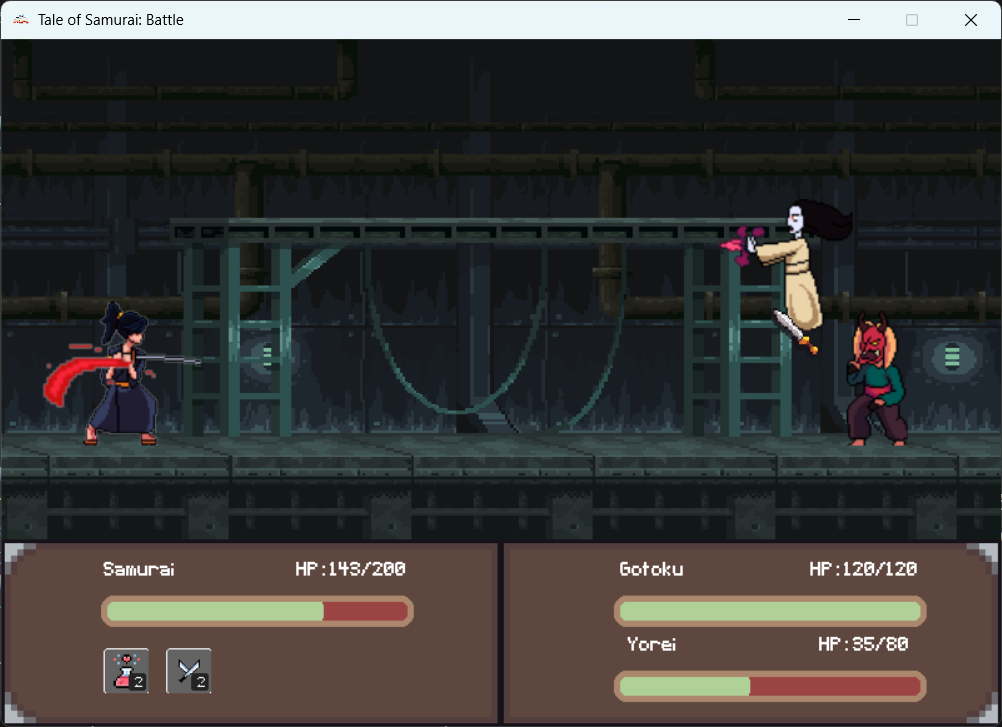
***Gambar 3.3.1.4*** *Animasi Samurai melakukan serangan*  *(Sumber : Dokumentasi Penulis)*

Ketika melakukan serangan, maka akan ada tampilan pedang yang menggores lawan serta gerakan dari Samurai seakan menghentaskan pedangnya. Karakter lawan juga akan mengalami pergerakan seakan terserang dan juga akan terjadi pengurangan HP.



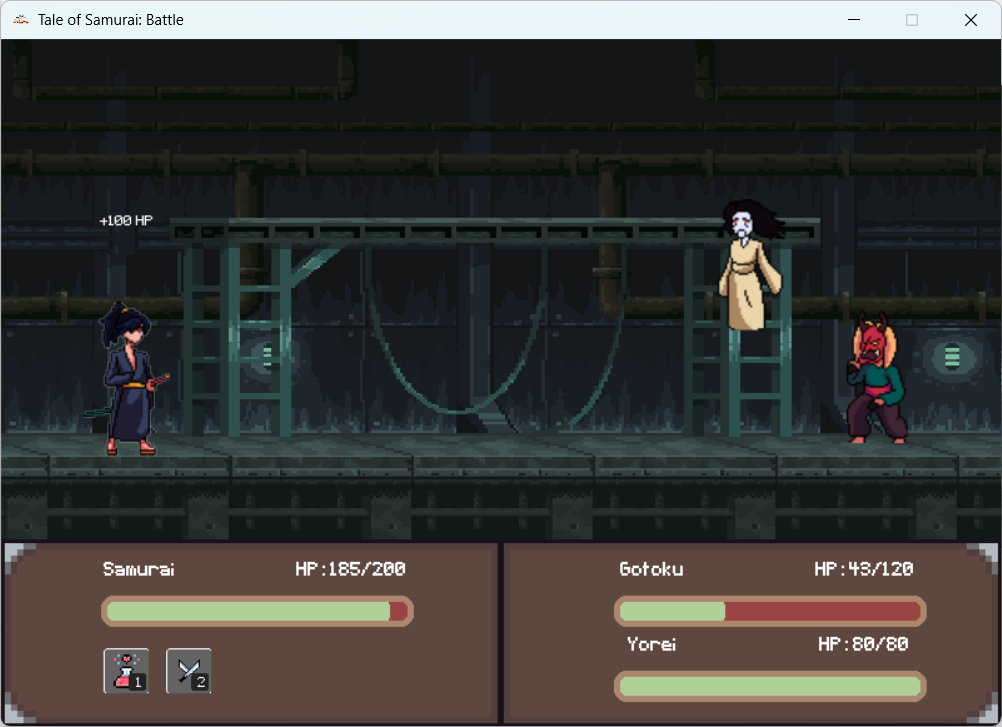
***Gambar 3.3.1.5*** *Animasi Samurai bertahan dari serangan Gotoku (Sumber : Dokumentasi Penulis)*

Setelah melakukan serangan, maka menjadi giliran untuk karakter lawan menyerang Samurai. Gotoku akan memulai serangan terlebih dahulu. Akan ada animasi gotoku ketika melakukan serangan, animasinya seakan Gotoku mencakar Samurai dan juga Samurai akan memiliki animasi untuk bertahan.



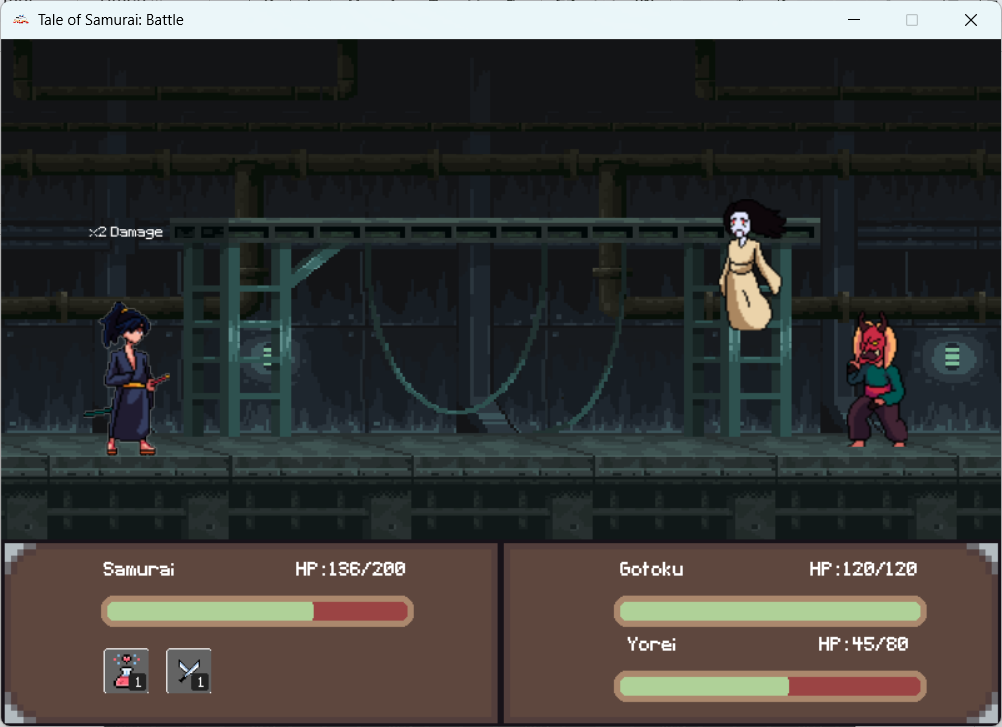
***Gambar 3.3.1.6*** *Animasi Samurai bertahan dari serangan Yorei* *(Sumber : Dokumentasi Penulis)*

Setelah Gotoku selesai melakukan serangan, maka akan menjadi giliran Yorei untuk melakukan serangan. Animasi dari Yorei berupa serangan magis yang memberikan efek darah pada Samurai.



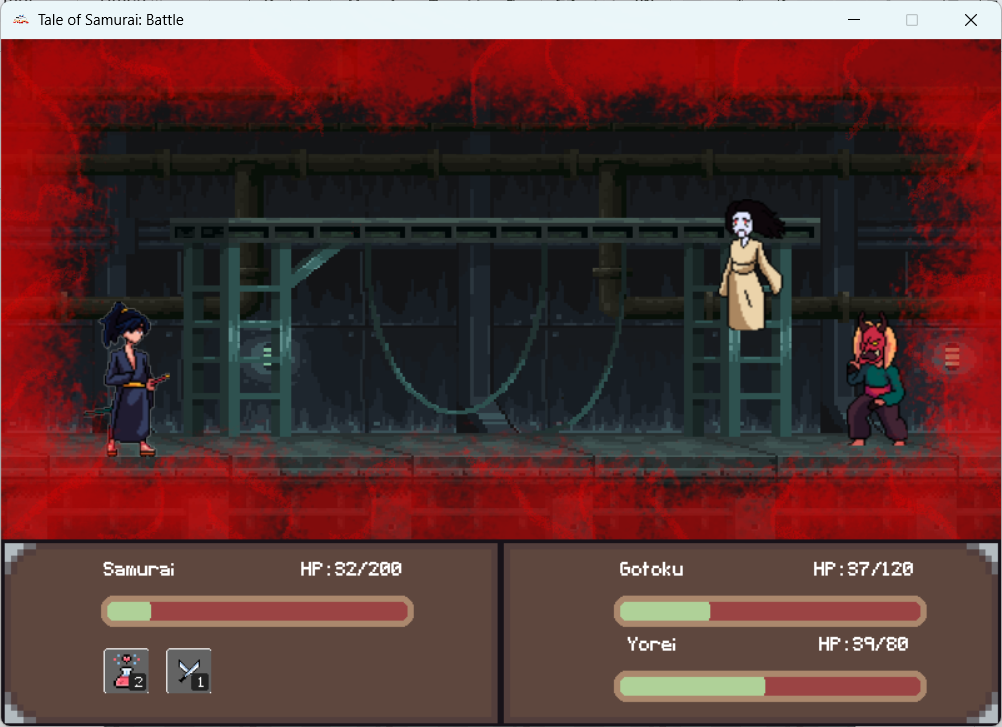
***Gambar 3.3.1.7*** *Samurai menggunakan charm heal (Sumber : Dokumentasi Penulis)*

Ketika memilih untuk menggunakan *charm* penambah HP, Samurai akan bisa mendapatkan HP sebanyak 100 HP. Penggunaan *charm* juga akan membuang giliran pemain.



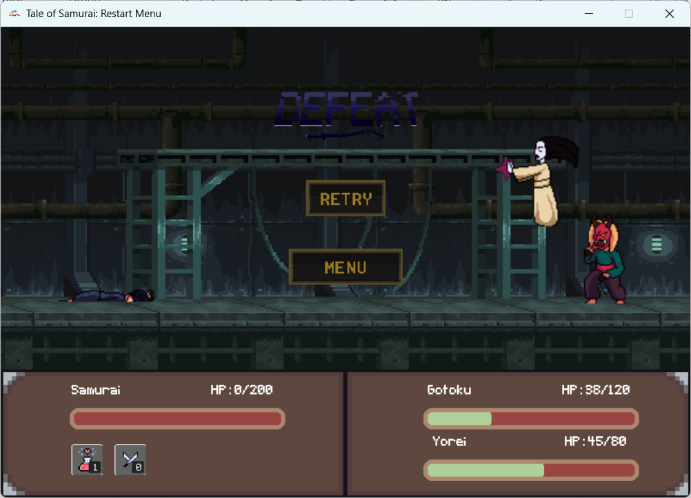
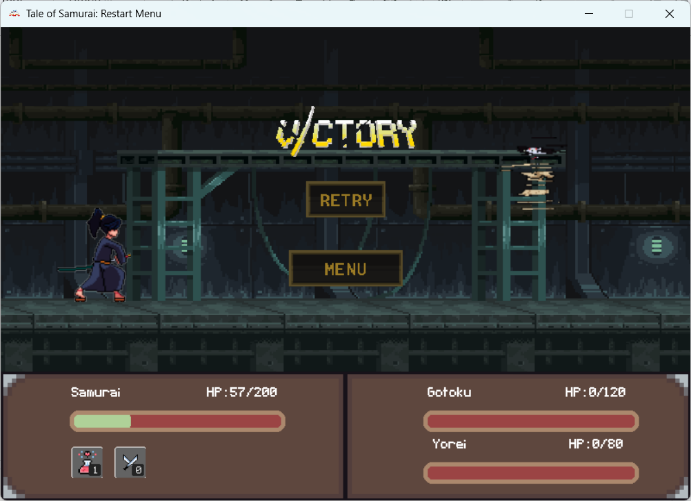
***Gambar 3.3.1.8*** *Samurai menggunakan charm double damage (Sumber : Dokumentasi Penulis)*

Samurai juga bisa menggunakan *charm* untuk meningkatkan *damage* sebanyak 2 kali. Penggunaan charm akan tersimpan hingga pemain melakukan serangan terhadap lawan. Jumlah serangan akan dikalikan dengan 2 ketika melakukan serangan pada lawan.



***Gambar 3.3.1.9*** *Efek halaman ketika darah sekarat (Sumber : Dokumentasi Penulis)*

Ketika HP dari Samurai kurang dari sama dengan 25% dari total HP, maka akan ada efek yang menunjukkan bahwa Samurai sedang sekarat. Pemain bisa menggunakan *charm* penambah HP untuk menghilangkan efek sekarat.

***Gambar 3.3.1.10*** *Tampilan halaman ketika menang dan kalah (Sumber : Dokumentasi Penulis)*

Permainan akan selesai ketika Samurai mati atau Samurai berhasil untuk membunuh kedua karakter lawan. Ketika Samurai mati, maka akan muncul notifikasi *Defeat* yang menandakan pemain kalah. Ketika Samurai berhasil untuk membunuh kedua karakter lawan, maka maka akan muncul notifikasi *Victory* yang menandakan bahwa pemain menang. Pemain juga bisa memilih opsi untuk kembali ke menu utama atau bermain ulang dengan menekan tombol *retry*.

## 3.4 Implementasi PBO

*Game Tale of Samurai* adalah *game* yang menerapkan konsep PBO. Sehingga tentunya banyak implementasi PBO yang digunakan pada *game* kami, berikut adalah implementasinya:

#### *Class* dan *Object*

Aplikasi yang menerapkan *OOP* tentunya menerapkan *class* dan *object* di dalamnya. *Tale of Samurai* menggunakkan berbagai elemen seperti *character, effect, healthbar, button* sebagai *class* dan sudah terdapat enam *class* yang digunakan. Berikut adalah salah satu contoh *class* yang sudah kami terapkan pada *game* kami, yaitu *class HealthBar()*.

|  |
| --- |
| class HealthBar():      def \_\_init\_\_(self, x, y, hp, max\_hp):          self.x = x          self.y = y          self.hp = hp          self.max\_hp = max\_hp          self.width = 250          self.height = 25          self.border\_radius = 10          self.health\_bar\_value = (self.hp / self.max\_hp) \* self.width        def draw(self, hp):          self.hp = hp          self.health\_bar\_value = (self.hp / self.max\_hp) \* self.width          bar\_surf = pygame.Surface((self.width, self.height), pygame.SRCALPHA)          pygame.draw.rect(bar\_surf, colors['red'], (0, 0, self.width, self.height), border\_radius=self.border\_radius)            pygame.draw.rect(bar\_surf, colors['green'], (0, 0, self.health\_bar\_value, self.height), border\_radius=self.border\_radius)          pygame.draw.rect(bar\_surf, colors['brown'], (0, 0, self.width, self.height), width=5, border\_radius=self.border\_radius)          screen.blit(bar\_surf, (self.x, self.y)) # 475 |

Dari setiap *class*, tentunya dibuat *object* yang akan digunakan secara langsung dalam *game*, sebagai contoh :

* 1. *Object Samurai, Gotoku,* dan *Yorei* yang dibuat dari *class Character*.
  2. *Object Claw, Slash,* dan *Blood* yang dibuat dari *class Effect*,
  3. *Object Health bar* untuk masing-masing *character* yang dibuat dari *class HealthBar*
  4. *Object button* yang digunakan dibuat dari *class button*.

#### *Inheritance*

*Tale of Samurai* memanfaatkan *inheritance* untuk mencegah penulisan ulang kode. *Class Sprite* yang merupakan *class* bawaan dari *pygame* mewarisi beberapa fungsi bawaan kepada *class Character* dan *class Effects* (*Hierarchical Inheritance*).

|  |
| --- |
| class Effect(pygame.sprite.Sprite):      def \_\_init\_\_(self, x, y, name):          self.name = name          self.x = x          self.y = y          self.animation\_list = []          self.animation\_index = 0          self.current\_time = pygame.time.get\_ticks()          self.scale\_ratio = 1.5            for i in range(1, 10):              if self.name == "Claw" or self.name == "Blood":                  self.scale\_ratio = 2.5              else:                  self.scale\_ratio = 0.8              img = pygame.image.load(f"assets/Effects/{self.name}/0{i}.png").convert\_alpha()              img = pygame.transform.rotozoom(img, 0, self.scale\_ratio)              self.animation\_list.append(img)            self.image = self.animation\_list[self.animation\_index]          self.rect = self.image.get\_rect()          self.rect.center = (x, y)        def animation(self):          animation\_delay = 100          self.image = self.animation\_list[int(self.animation\_index)]            if pygame.time.get\_ticks() - self.current\_time >= animation\_delay:              self.current\_time = pygame.time.get\_ticks()              self.animation\_index += 1        def draw(self):          screen.blit(self.image, self.rect)        def update(self):          self.animation() |

*Class AbsHealthBar* juga mewarisi beberapa variabel kepada *class HealthBar* (*Single Inheritance*) (Hasil *coding* akan di bahas pada sub bab selanjutnya).

#### *Abstraction*

*Abstraction* atau abstraksi digunakan untuk menyediakan *blueprint* dari suatu *class. Abstraction* membantu mendefinisikan atribut dan fungsi yang harus dimiliki oleh *class* turunan. Sebagai contoh, *Class AbshealthBar* menyediakan beberapa variabel dasar untuk *class HealthBar,* yang kemudian variabel tersebut harus dimiliki oleh *class Healthbar.*

|  |
| --- |
| class AbsHealthBar(ABC):      @property      @abstractmethod      def hp(self):          pass      def max\_hp (self):          pass  class HealthBar(AbsHealthBar):      def \_\_init\_\_(self, x, y, hp, max\_hp):          self.x = x          self.y = y          self.hp = hp          self.max\_hp = max\_hp          self.width = 250          self.height = 25          self.border\_radius = 10          self.health\_bar\_value = (self.hp / self.max\_hp) \* self.width        def draw(self, hp):          self.hp = hp          self.health\_bar\_value = (self.hp / self.max\_hp) \* self.width          bar\_surf = pygame.Surface((self.width, self.height), pygame.SRCALPHA)          pygame.draw.rect(bar\_surf, colors['red'], (0, 0, self.width, self.height), border\_radius=self.border\_radius)            pygame.draw.rect(bar\_surf, colors['green'], (0, 0, self.health\_bar\_value, self.height), border\_radius=self.border\_radius)          pygame.draw.rect(bar\_surf, colors['brown'], (0, 0, self.width, self.height), width=5, border\_radius=self.border\_radius)          screen.blit(bar\_surf, (self.x, self.y)) # 475 |

#### *Polymorphism*

*Class* *Character* dan *Effect* dibuat secara fleksibel sehingga dapat digunakan secara berbeda-beda oleh masing-masing *object* sesuai kebutuhan. Sebagai contoh, *class Character* dapat digunakan untuk *object Samurai, Yorei* dan *Gotoku.*

|  |
| --- |
| samurai = Character(100, 270, 'Samurai', 200, 40, charms)      samurai\_health\_bar = HealthBar(80, (screen\_height + bottom\_panel\_height / 2) - 30, samurai.hp, samurai.max\_hp)      gotoku = Character(screen\_width - 100, 270, 'Gotoku', 120, 25, None)      gotoku\_health\_bar = HealthBar(screen\_width - 310, (screen\_height + bottom\_panel\_height / 2) - 30, gotoku.hp, gotoku.max\_hp)      yorei = Character(screen\_width - 200, 180, 'Yorei', 80, 30, None)      yorei\_health\_bar = HealthBar(screen\_width - 310, (screen\_height + bottom\_panel\_height / 2) + 30, yorei.hp, yorei.max\_hp)      enemies = []      enemies.append(gotoku)      enemies.append(yorei) |

*Class Effect* dapat digunakan untuk *object Slash, Blood*, dan *Claw*.

|  |
| --- |
| samurai.attack(target)  active\_effects.append(Effect(target.rect.centerx, target.rect.centery, "Slash"))  if enemy.name == "Yorei":  active\_effects.append(Effect(samurai.rect.centerx, samurai.rect.centery, "Blood"))  elif enemy.name == "Gotoku":  active\_effects.append(Effect(samurai.rect.centerx, samurai.rect.centery, "Claw")) |

#### *Relationship*

Beberapa *class* yang saling terhubung satu dengan yang lain memiliki tingkatan *relationship* nya masing-masing. Sebagai contoh, *class HealthBar* yang merupakan komposisi dari *class Character* karena *class HealthBar* tidak dapat berdiri sendiri. *Class FloatingText* dan *Class Effect* merupakan agregasi dari *class Character*.

# BAB IV

# PENUTUP

## 4.1 Kesimpulan

*Tale of Samurai* adalah *game turn-based RPG* yang dibangun menggunakkan bahasa pemrograman python dengan bantuan *library pygame*.Dengan penerapan dan implementasi PBO, pengembangan *game* menjadi lebih terstruktur, efisien dan mudah dikembangkan.

Penggunaan PBO memungkinkan modularitas dalam desain game, seperti pemisahan *class* untuk *character*, *effect*, elemen *GUI*. Hal ini tidak hanya meningkatkan efisiensi pengembangan tetapi juga mempermudah proses pembaruan atau penambahan fitur. Kombinasi antara **Python** sebagai bahasa yang fleksibel dan ***pygame*** sebagai library yang kuat mendukung implementasi elemen-elemen utama dalam game, seperti animasi, sound effect, dan interaksi pengguna.

Sehingga, ***Tale of Samurai*** tidak hanya menawarkan pengalaman bermain yang menarik melalui mekanisme turn-based RPG, tetapi juga menjadi bukti bagaimana penerapan konsep pemrograman yang baik dapat menghasilkan aplikasi yang optimal.

## 4.2 Saran

Pengembangan *game Tale of Samurai* memiliki potensi yang besar untuk dikembangkan, sehingga kedepannya dapat dilakukan perkembangan lebih lagi. Contohnya dengan menambahkan *elemen-elemen GUI* yang lebih dinamis dan interaktif, penambahan *stage-stage* baru dengan latar belakang dan *character* lain yang semakin memperkaya *game*.

Setiap stage dapat dirancang untuk memiliki misi atau tantangan khusus yang membuat alur permainan semakin seru. Alur cerita yang menarik juga dapat ditambahkan, misalnya dengan narasi yang memperdalam latar belakang karakter utama dan konfliknya dengan musuh-musuh, sehingga pemain merasa lebih terhubung dengan cerita.