3. ANALISIS DAN DESAIN SISTEM

3.1 Analisis Masalah dan Kebutuhan

Pembuatan *Balancing Artificiall Intelligence* ini terinspirasi dari metode yang sebelumnya sudah ditemukan oleh Emmett Tomai dan Roberto Flores yang tercatat di dalam jurnal yang berjudul "*Adapting In-Game Agent Behavior by Observation of Players Using Learning Behavior Trees".* Penelitian tersebut membuat sebuah *AI* yang dapat beradaptasi dari jejak yang dilalui oleh *Player,* yang dinamai dengan *Learning Behavior Tree.* Metode ini menggunakan *Behavior Tree* yang di-set di dalam *AI* yang nantinya diselaraskan dengan *playback* dari jejak kegiatan *player, AI* akan diupdate yaitu dengan mendeteksi apakah yang sedang dilakukan oleh *player*, jika terdapat kesamaan maka *AI* akan di set untuk memiliki delay yang lebih lama daripada *player* dari setiap kegiatan yang dilakukan oleh *AI* maupun *player,* dan jika tidak sama maka *AI* akan melakukan kegiatannya seperti biasanya tanpa diberikan delay apapun. *AI* yang akan dibuat ini tidak sepenuhnya mirip dengan jurnal tersebut, *AI* yang dibuat ini akan menggunakan *real time delay* terhadap setiap aksinya jika *Score AI* terlalu melampaui dari *Score Player*. Serta, *Balancing Artificiall Intelligence* ini akan dirancang untuk tidak memerlukan *training* sebelum dapat digunakan sebagai lawan *player*.

Penelitian ini menggunakan *Knapsack Problem* dan *Traveling Salesman Problem* karena kedua permasalahan ini adalah masalah yang sering muncul dalam contoh pembelajaran algoritma. Kedua masalah ini sulit karena solusi terbaiknya itu tergantung pada banyak variabel dan kombinasi yang sangat besar, terutama ketika jumlah item atau kota yang ada sangat banyak. *Player* dapat mencoba algoritma apapun dalam memecahkan permasalahan di dalam Game ini, sehingga dapat menentukan algoritma mana yang baik untuk menyelesaikannya, serta belajar bagaimana menyelesaikan permasalahan ini dengan algoritma tersebut jika belum memahami bagaimana cara berjalannya algoritma - algoritma tersebut. Sama halnya dengan *Traveling Salesman Problem,* player dapat belajar bagaimana mengatur jejak yang terbaik untuk mendapatkan value atau dapat menambang sebanyak yang dimau sekali jalan, dan sebagainya. Selain itu, mengembangkan algoritma efisien untuk kedua masalah ini juga merupakan tantangan tersendiri. Siswa mungkin mengalami kesulitan dalam memahami dan menerapkan teknik-teknik seperti algoritma greedy,

serta memilih strategi yang tepat untuk menyelesaikan masalah dengan waktu eksekusi yang wajar juga merupakan aspek penting yang perlu dipelajari. Kesulitan lainnya adalah dalam memahami relevansi kedua masalah ini dengan dunia nyata dan aplikasi praktisnya. Siswa mungkin merasa kesulitan dalam melihat bagaimana *Knapsack problem* dapat diterapkan dalam kehidupan nyata seperti perencanaan sumber daya atau manajemen *inventory*, serta bagaimana *Travelling Salesman Problem* dapat digunakan dalam perutean kendaraan atau rute pengiriman.

Visualisasi dalam bentuk *game* dapat membantu untuk memperjelas masalah dan memudahkan pemahaman tentang solusi terbaiknya. Pembelajaran menggunakan game adalah pendekatan yang populer dalam studi informatika, *game* dapat menjadi alat yang efektif untuk mengajarkan konsep-konsep yang kompleks, dan memperkuat keterampilan. Visualisasi menggunakan *game* juga dapat digunakan untuk mengajarkan algoritma pemrograman, struktur data, *artificial intelligence*, serta meningkatkan keterampilan pemecahan masalah dan kreativitas siswa. Dalam *game*, *player* dapat melihat langsung bagaimana keputusan yang diambil mempengaruhi nilai total atau jarak yang ditempuh, serta bagaimana memilih solusi yang salah dapat menghasilkan hasil yang buruk. *Game* juga dapat memberikan *feedback* visual tentang perubahan nilai dan hasil pemilihan, sehingga dapat membantu mempercepat pemahaman dan membuat belajar lebih menyenangkan.

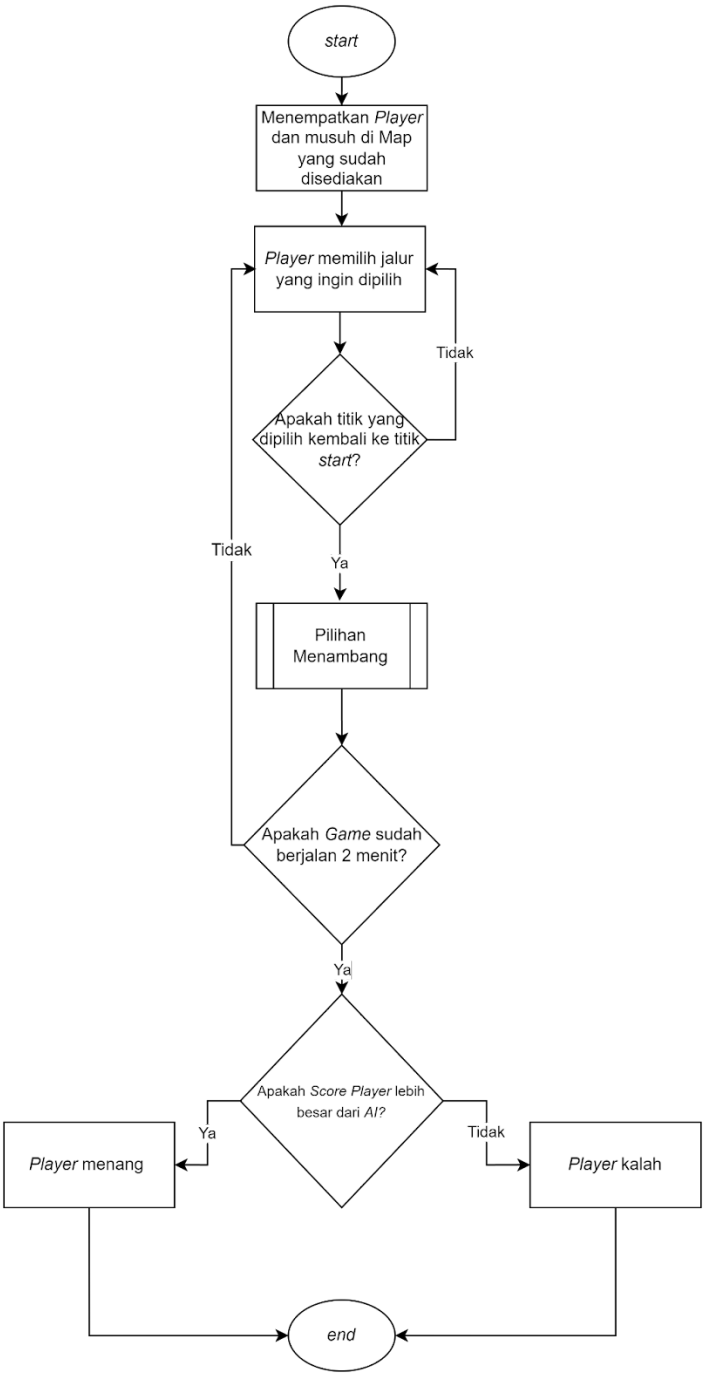
3.2 Desain Sistem

3.2.1 Desain Story dan Gameplay

Disuatu hari yang biasa biasa saja, terdapat seorang bounty hunter yang sedang menelusuri tempat mining yang sudah ditinggalkan, disaat sudah mencapai cukup dalam, ternyata ada gempa bumi yang menyebabkan tempat tersebut runtuh dan membuat Alex tertimbun oleh reruntuhan tempat mining tersebut. Setelah beberapa jam kemudian, Alex terbangun di sebuah tempat, di depan hadapannya terdapat sosok zombie yang bisa berbicara, zombie tersebut akhirnya menyambut calon makanannya dengan sangat riang, Alex pun berlari mencari jalan keluar, dan ternyata dia baru menyadari bahwa tempat yang Ia berada ini berada di ruang dan dimensi yang lainnya, akhirnya Zombie berkata bahwa tempat ini adalah Mind Miner. Jika ingin keluar dari tempat ini maka Alex harus mengalahkan Zombie. Akan tetapi, jika Alex kalah, maka Zombie akan memakan otak Alex. Akhirnya pun Alex setuju dengan Zombie dan menerima tantangannya tersebut.

Cara *Player* dan musuh mengelilingi tempat ini adalah memilih titik - titik pada map yang sudah disediakan, *Player* dan musuh harus dengan hati-hati memilih jalur yang menurutnya paling bagus, setiap titik tersebut harus kembali ke titik *Start*, jika sudah kembali ke titik *Start*, maka karakter dapat langsung jalan. Pada beberapa titik yang telah dipilih, akan ada tempat untuk menambang. Pemilihan titik - titik pada map akan menggunakan *Mouse* klik kiri, dan klik kiri lagi pada titik yang sudah dipilih untuk tidak jadi memilih titik tersebut. Terdapat beberapa *sistem* di dalam *Game* ini, seperti *backpack* yang menggunakan *Knapsack Problem*, lalu terdapat juga *Travelling Salesman Problem* yang digunakan sebagai sistem untuk mengelilingi daerah yang dipilih, serta terdapat sistem yang selama beberapa menit sekali akan membuka daerah baru untuk dapat dikunjungi.

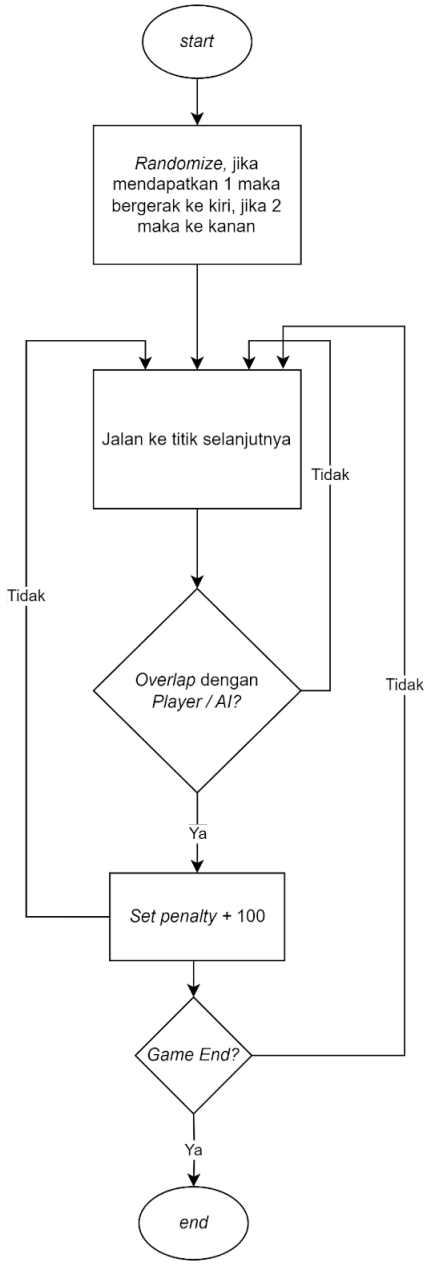
Game ini akan berlangsung selama 2 menit, dan pemenangnya adalah yang memiliki *Value* tertinggi dalam barang-barang yang dikumpulkan.



Gambar 3.1 *Flowchart* Alur *Gameplay*

3.2.2 Desain sistem *AI Patrol*

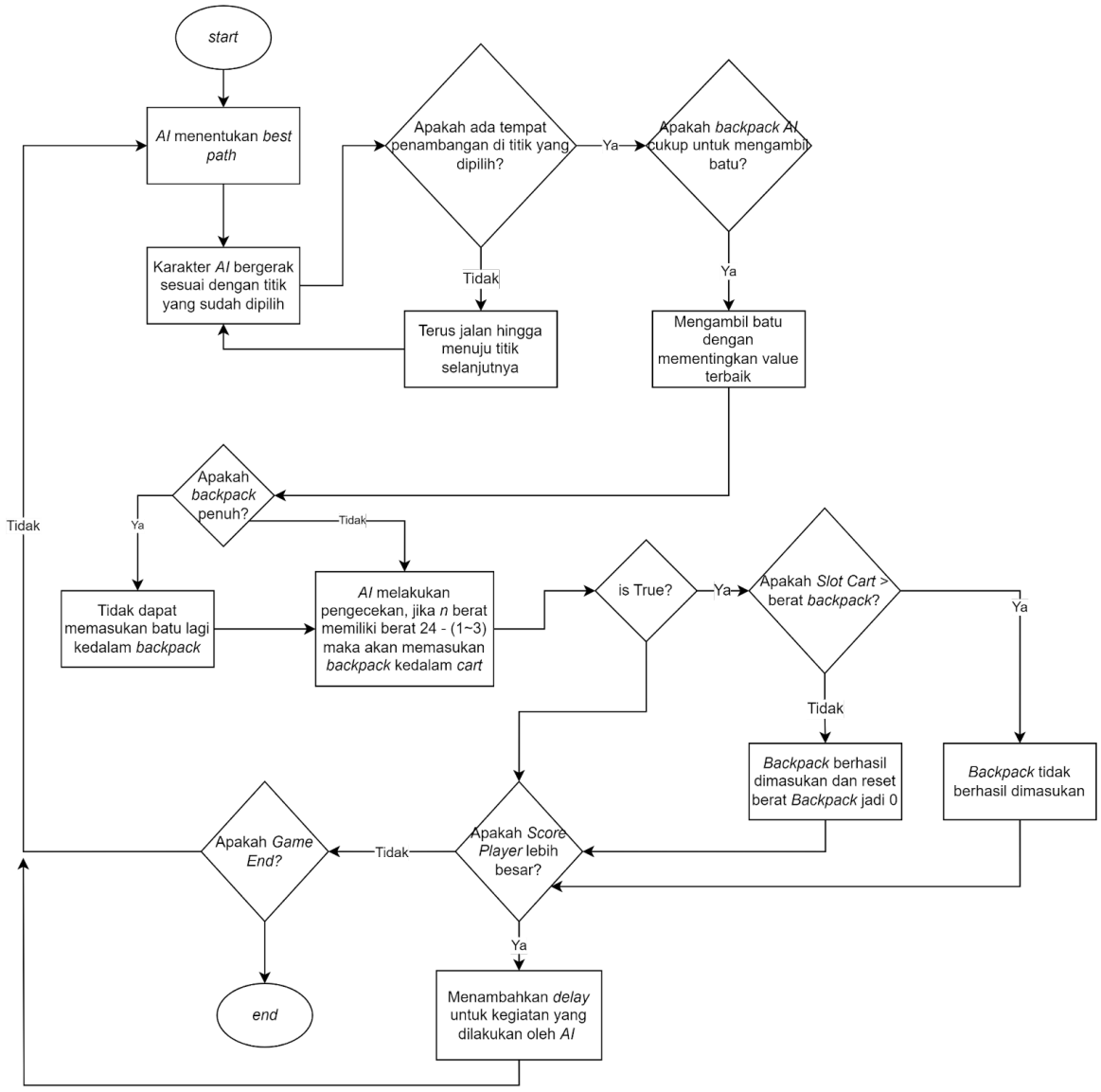
AI ini akan berkeliling di batas antara daerah *Player* dan *AI Competitor. AI* ini akan memberikan penalti jika saling bertabrakan dengan *Player* maupun *AI Competitor. AI* ini tidak akan berhenti bergerak selama waktu masih bergulir.



Gambar 3.2 *Flowchart* Sistem *AI Patrol*

3.2.3 Desain sistem Balancing Artificial Intelligence

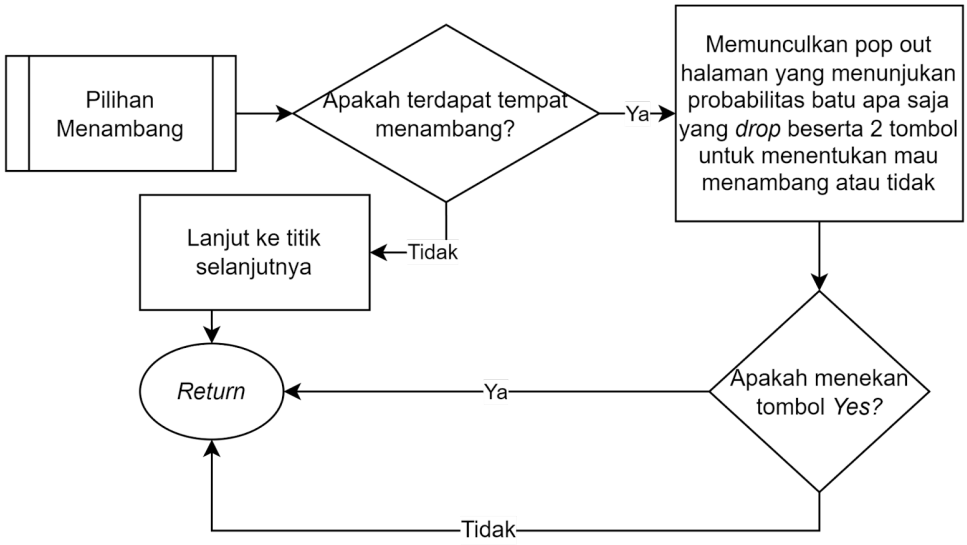
Artificial Intelligence yang digunakan pada *Game* ini adalah *Balancing Artificial Intelligence*. *AI* ini akan bekerja untuk melakukan *Balancing* terhadap permainan agar lebih balance lagi. *AI* akan diberikan "punishment" jika Score *AI* lebih besar daripada *Player*. Sistem *AI* ini sama seperti *player* yang dimana AI dapat memilih Path terbaik, terdapat priority rules, rules pertama adalah jika titik selanjutnya merupakan jalan keluar, maka akan langsung memilih titik tersebut. Jika bukan, maka akan memilih priority rules kedua yaitu jika titik selanjutnya memiliki tempat Mining maka akan langsung memilih titik tersebut. Jika kedua Rules tersebut masih tidak terpenuhi, maka akan memilih secara random titik yang akan dituju. *AI* dapat melakukan pengecekan disaat Score dari Player lebih sedikit dari *AI*, maka akan ditambah delay untuk setiap kegiatan yang dilakukan oleh *AI*, dan akan kembali seperti semula jika Player lebih unggul. *AI* juga melakukan pengecekan jika berat Backpack adalah 21 kg hingga 24 kg maka akan melakukan pengecekan pada *cart* dan *backpack* sistem.



Gambar 3.3 *Flowchart* Sistem *Balancing Artificial Intelligence*

3.2.4 Desain Sistem Pilihan Menambang

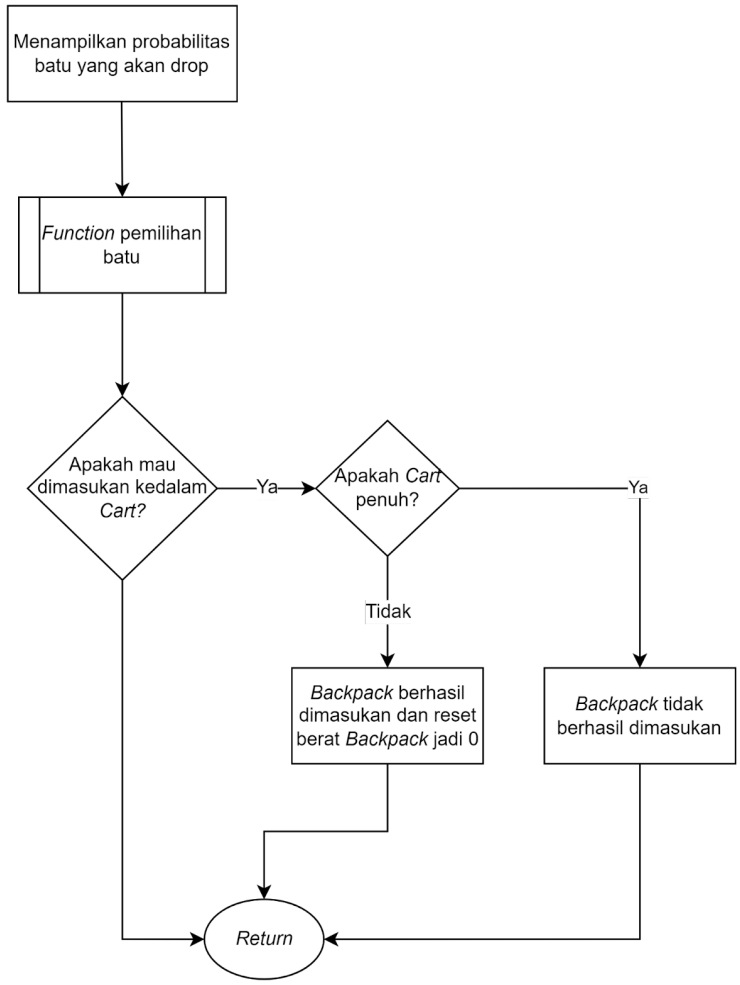
Pemilihan titik - titik pada map akan menggunakan *Mouse* klik kiri, dan klik kiri lagi pada titik yang sudah dipilih untuk tidak jadi memilih titik tersebut. Ketika sampai pada titik yang dimana terdapat tempat untuk menambang, akan memunculkan pilihan 2 button *Yes* dan *No* untuk memilih apakah mau menambang di sana setelah mendapatkan *info* batu apa saja yang bisa drop di tempat itu. *Player* akan diberikan waktu selama 10 detik untuk menentukan batu mana saja yang mau diambil.



Gambar 3.4 *Flowchart* Sistem Pilihan Menambang

3.2.5 Desain Sistem Menambang

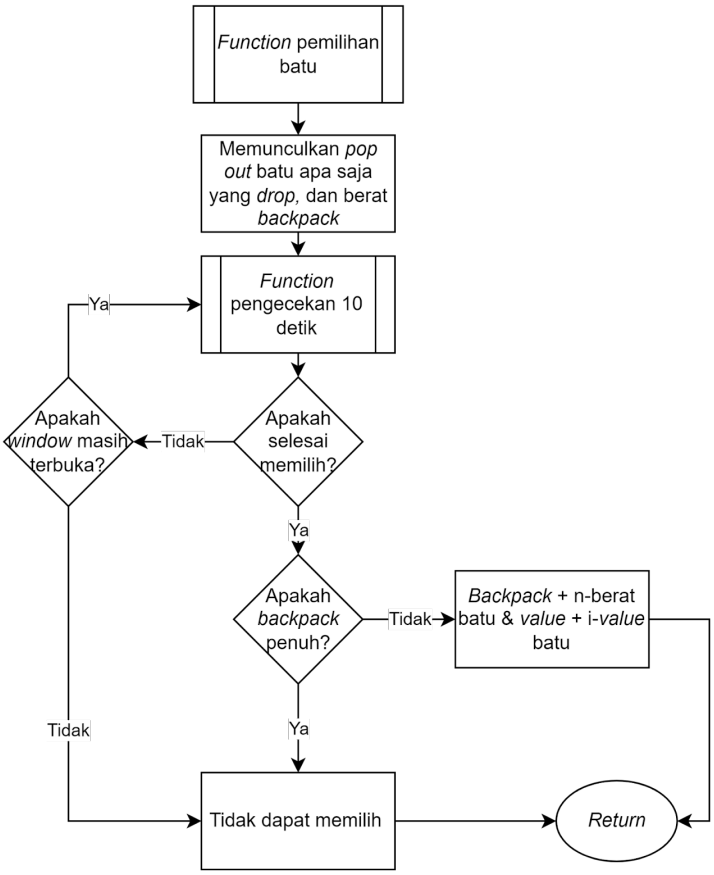
Player dan musuh akan memiliki *Cart* yang berfungsi untuk membawa batu - batu berharga yang sudah ditambang, *Cart* tersebut memiliki maksimal beban yaitu 100. *Player* dan musuh juga memiliki *Backpack* untuk mengangkut batu berharga yang *drop* setelah menambang (*Cart* merupakan kumpulan dari *backpack*). Sistem menambang terdapat beberapa *step*, yaitu menampilkan probabilitas dari batu yang akan *drop* pada tempat itu, jika setuju untuk menambang di tempat itu, maka dilanjutkan ke pemilihan batu yang drop ditempat tersebut, setelah memilih batu batu tersebut, memunculkan apakah *player* atau *AI* mau untuk memasukan kedalam *cart* atau tidak.



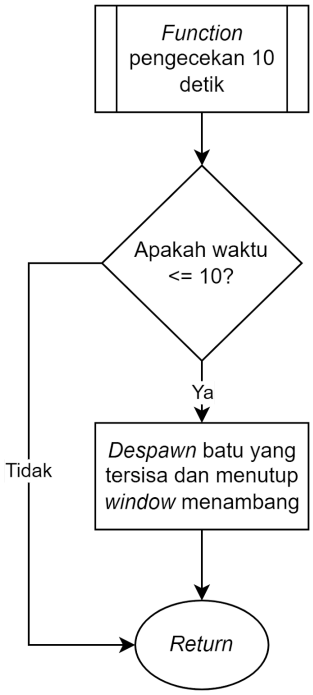
Gambar 3.5 *Flowchart* Sistem Menambang

3.2.6 Desain sistem Pemilihan Batu

Sistem ini akan mengecek apakah *backpack* dapat menerima batu lagi atau tidak.



Gambar 3.6 *Flowchart* Sistem Pemilihan Batu

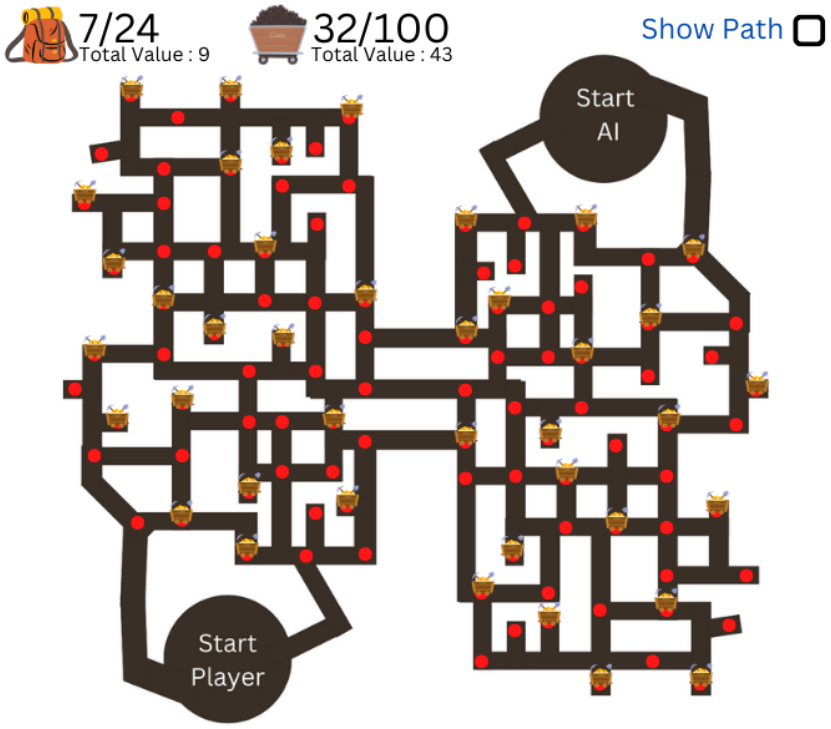


Gambar 3.7 *Flowchart* Sistem Fungsi Pengecekan 10 Detik

3.3 Desain Tampilan User Interface

3.3.1 User Interface Map

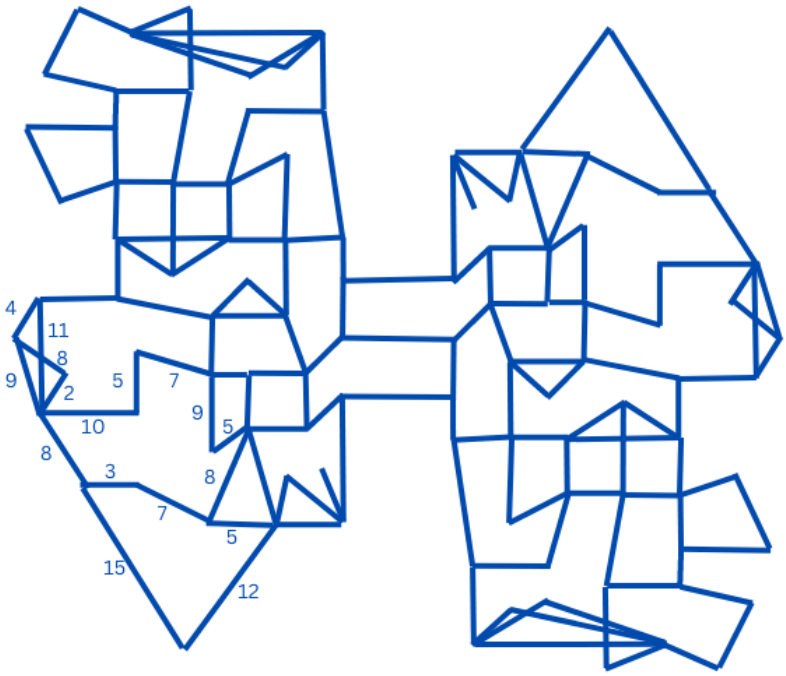
Pada Gambar 3.8 menggambarkan tempat start *Player* dan AI berbeda, *player* berada di bawah sebelah kiri sedangkan *AI* berada di atas sebelah kanan.



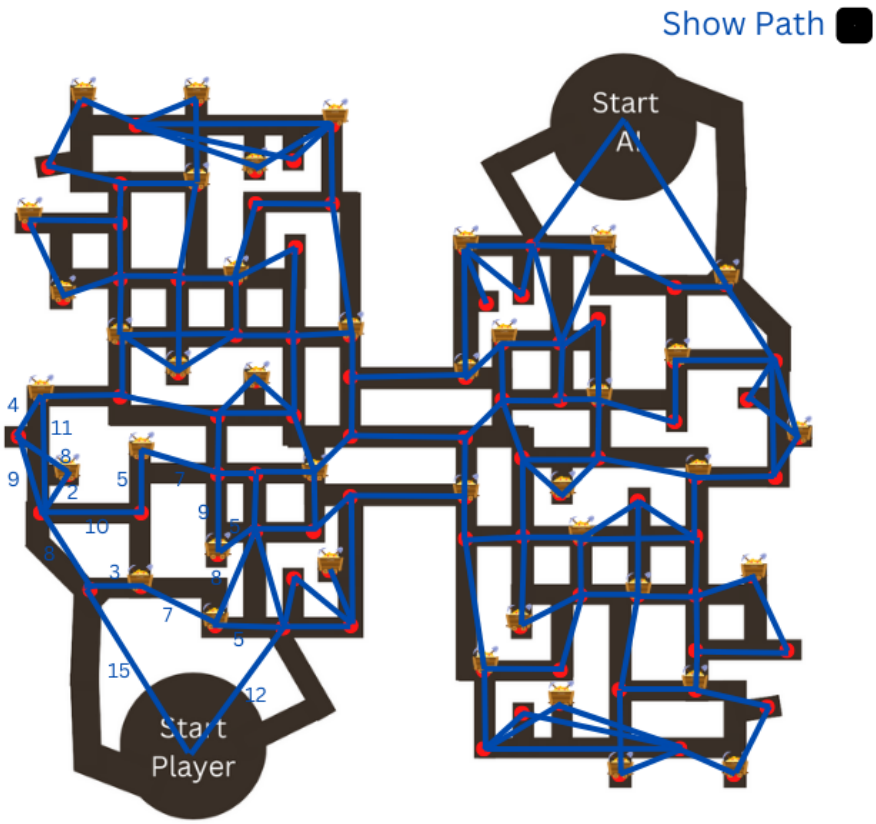
Gambar 3.8 Desain *User Interface Full Map*

3.3.2 User Interface Path

Pada Gambar 3.10 menggambarkan *path* yang digunakan untuk menandakan jarak antar titik 1 dengan yang lainnya, yang memakai konsep *Travelling Salesman Problem.* Gambar 3.11 menggambarkan *map* jika menekan Spasi pada *keyboard* akan memunculkan *path*.



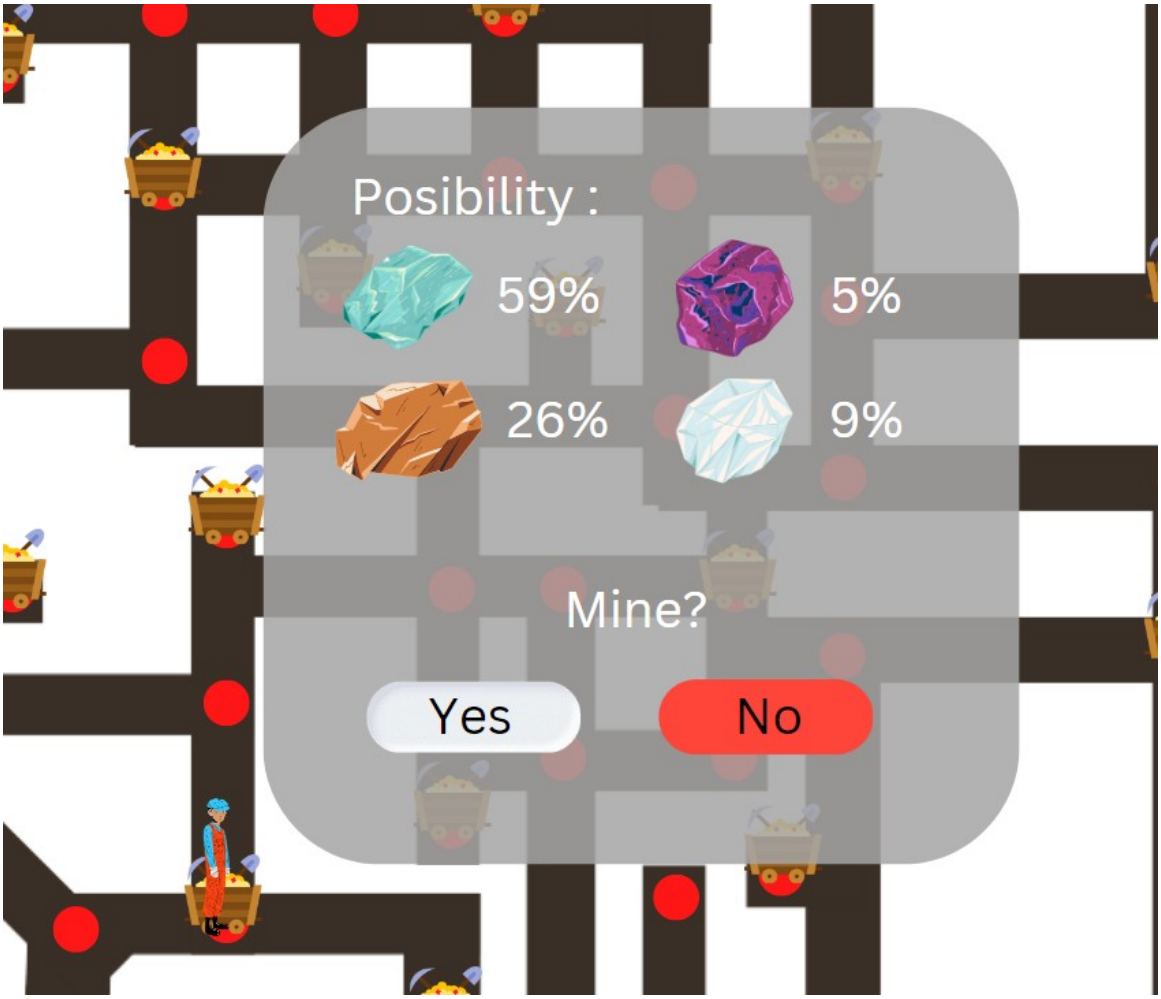
Gambar 3.9 Desain *User Interface Path*



Gambar 3.10 Desain *User Interface Map* dan *Path* digabungkan

3.3.3 User Interface Pop Out Menambang

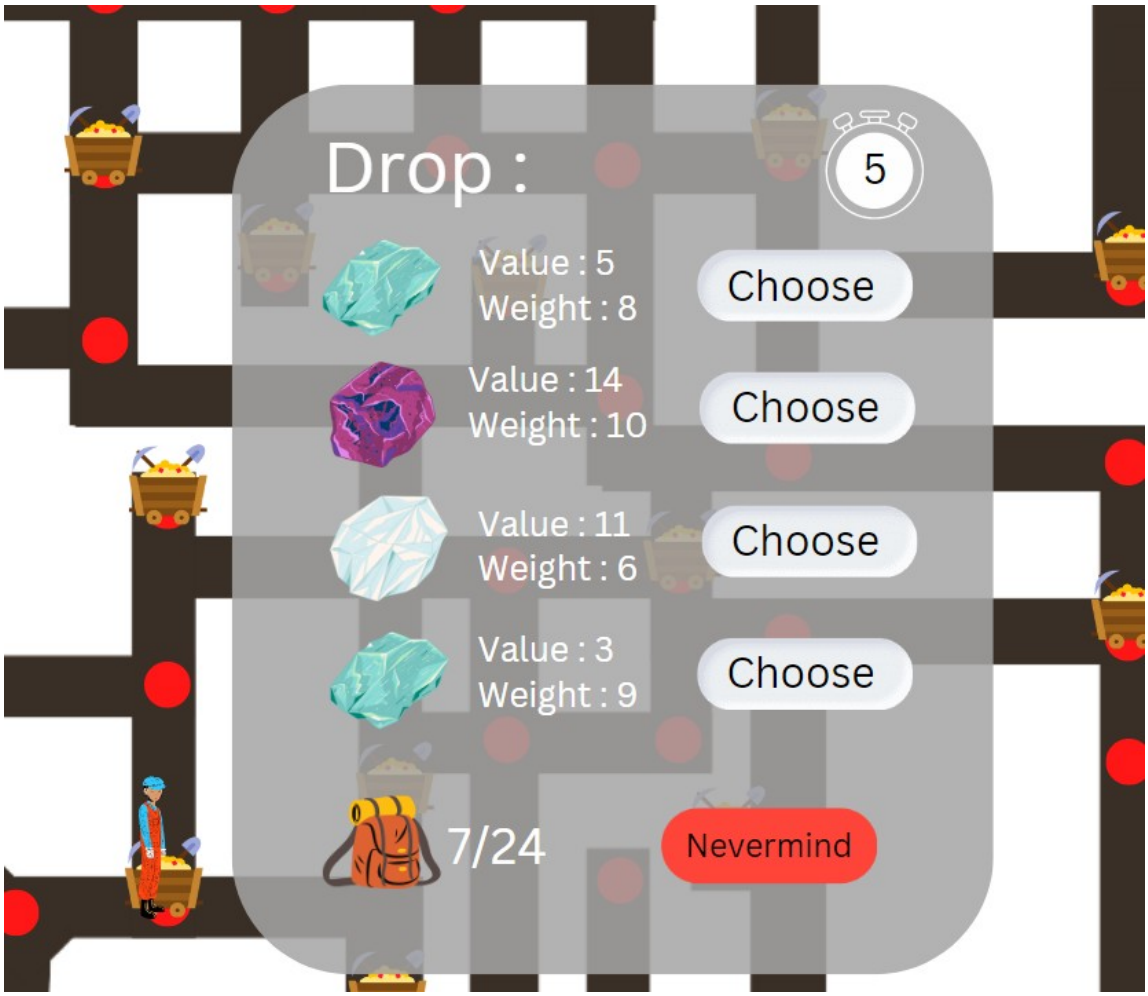
Disaat player tiba di tempat menambang, maka akan memunculkan *pop out* yang berisi persentase batu yang akan *drop*.



Gambar 3.11 Desain *Pop Out* Persentase Menambang

3.3.4 User Interface Pop Out Memilih Batu

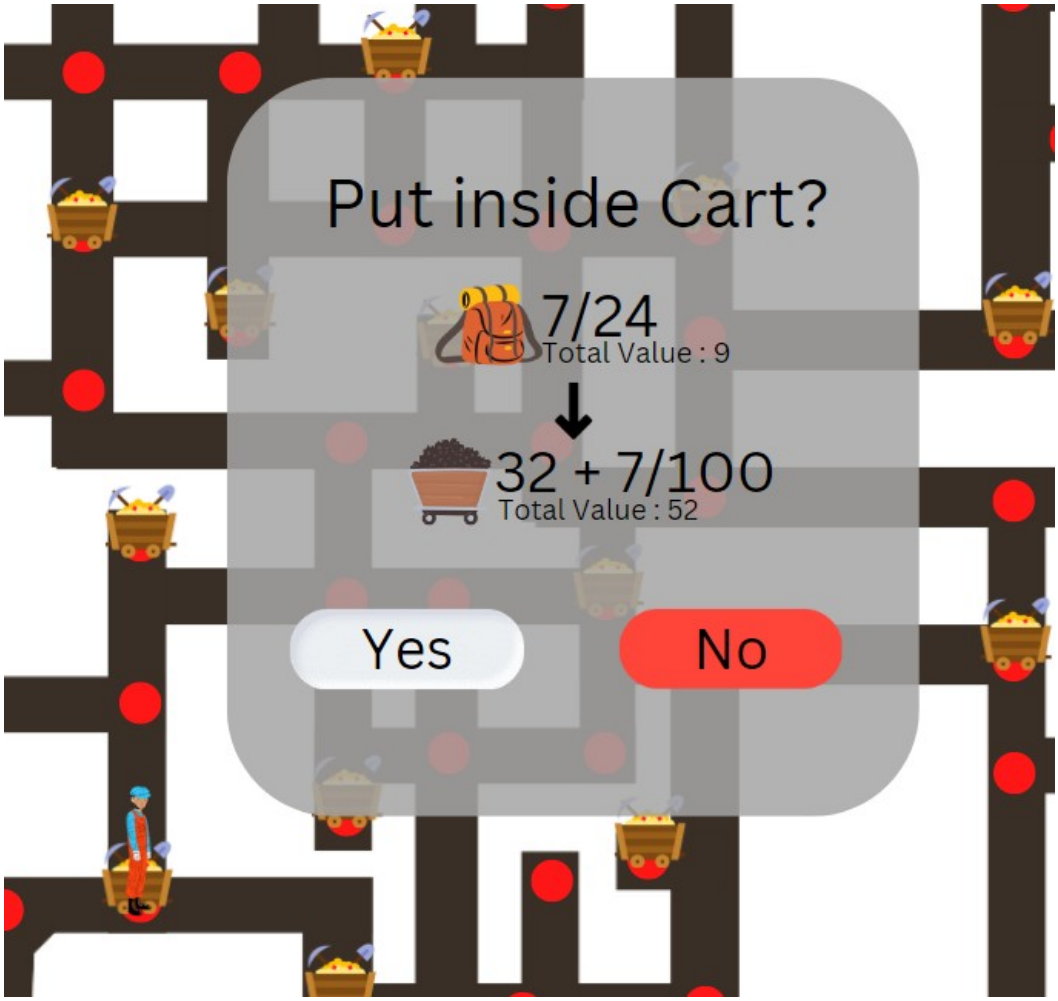
Setelah memilih Yes, maka akan langsung channeling selama 3 detik dan setelah itu memunculkan kembali pop out yang berisikan batu apa saja yang drop dan berat backpack serta total value didalam backpack.



Gambar 3.12 Desain *Pop Out* Pemilihan Batu

3.3.5 User Interface Memasukan Dalam Cart

Setelah memilih atau tidaknya batu yang sudah drop, akan ditujukan pada pop out yang berisikan pilihan untuk memasukan backpack ke dalam cart.



Gambar 3.13 Desain *Pop Out* Memasukan ke dalam *Cart*

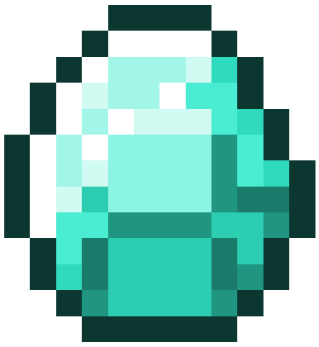
3.3.6 Tampilan Desain Character

Gambar 3.14 hingga 3.15 menggambarkan desain karakter yang akan dipakai dalam pembuatan *Game* ini.

Gambar 3.14 Desain Karakter *AI* Kompetitor

Gambar 3.15 Desain Karakter *Player*

3.3.7 Tampilan Desain Batu Berharga

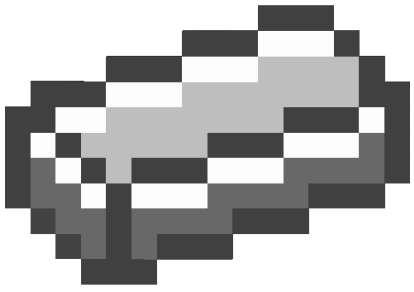


Gambar 3.16 Desain Diamond

Sumber : Diamond. (2010). Minecraft Wiki. https://minecraft-archive.fandom.com/wiki/Diamond

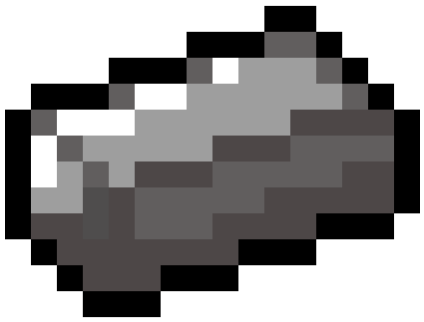
Gambar 3.17 Desain Gold

Sumber : Gold Ingot. (2010). Minecraft Wiki. https://minecraft-archive.fandom.com/wiki/Gold_Ingot



Gambar 3.18 Desain Silver

Sumber : Iron Ingot. (2010). Minecraft Wiki. https://minecraft-archive.fandom.com/wiki/Iron_Ingot



Gambar 3.19 Desain Vibranium

Sumber : Netherite Ingot. (2020). Minecraft Wiki.

https://minecraft-archive.fandom.com/wiki/Netherite_Ingot