BAB III
METODE PENELITIAN

3.1 Metode Penelitian

Metode penelitian yang digunakan adalah dengan cara mencoba menjalankan dan memainkan permainan dengan berbagai tingkat kesulitan yang ada, kemudian dilakukan dua pengujian yaitu pengujian antarmuka program dan kesesuaian kecerdasan buatan dengan cara sebagai berikut:

1. Penguji mencoba semua elemen antarmuka yang ada pada program. Dari awal pertama program dibuka hingga program ditutup tidak ada kesalahan aksi ataupun *error* yang terjadi.
2. Penguji memainkan permainan *Tic Tac Toe* pada program, kemudian menilai apakah tingkat kesulitan yang ada telah sesuai.

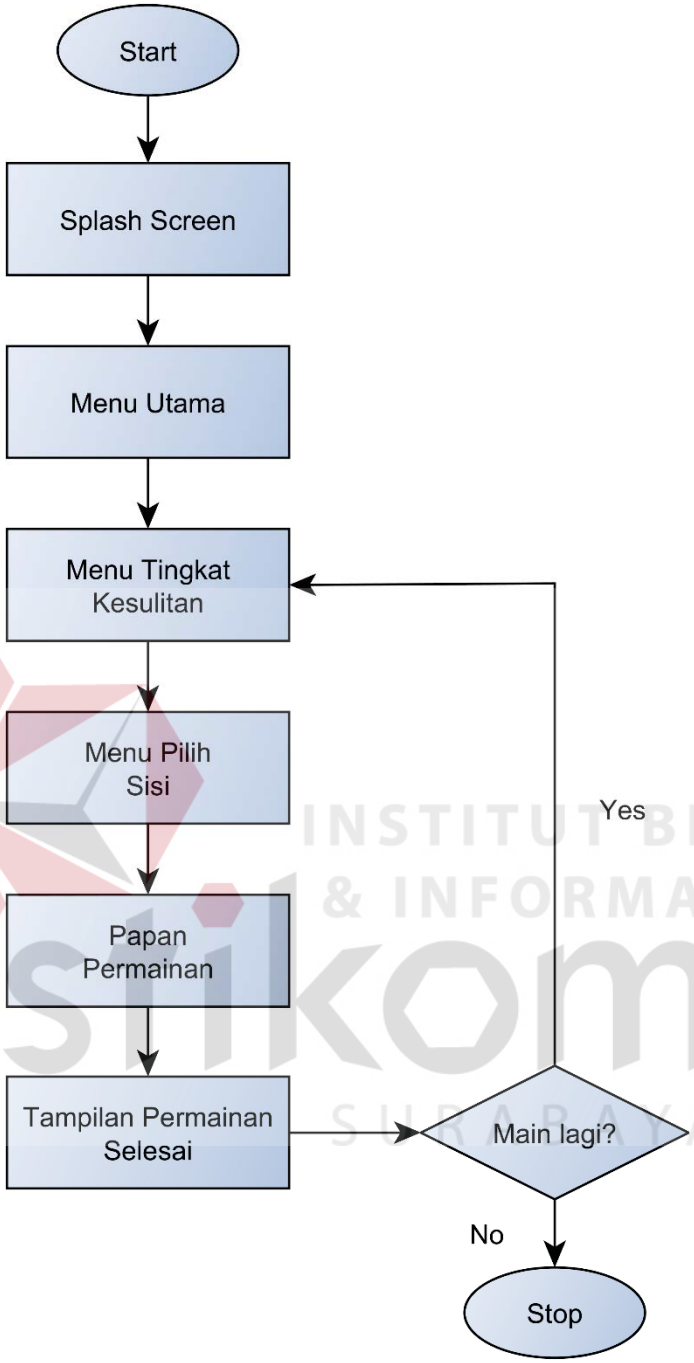
Apabila masih terdapat kesalahan pada aplikasi yang telah dibuat, maka dilakukan perbaikan atau troubleshooting pada aplikasi sehingga dapat menunjukkan hasil yang diharapkan.

3.2 Perancangan Aplikasi

Perancangan aplikasi dibagi menjadi beberapa bagian pengerjaan yaitu sebagai berikut:

3.2.1 Perancangan Antarmuka

Perancangan antarmuka dititikberatkan pada ketepatan sasaran pilihan menu. Adapun gambaran umum untuk rancangan menu dengan *flowchart*.



Gambar 3.1 *Flowchart* Rancangan Menu Aplikasi

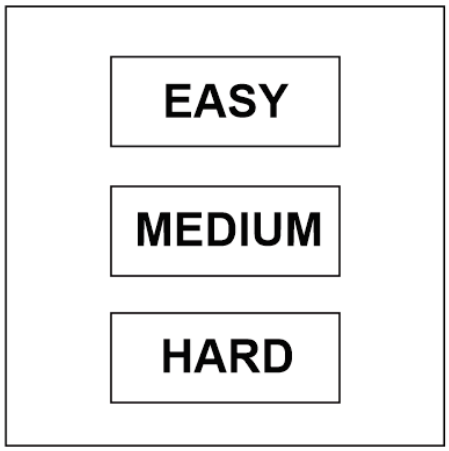
Saat aplikasi dijalankan, tampilan *Splash Screen* akan muncul sesaat, kemudian Menu Utama akan muncul.

Gambar 3.2 Rancangan Tampilan *Splash Screen*

Menu Utama akan memberikan dua pilihan, yaitu mulai bermain yang diwakili dengan tombol "Play" dan keluar dari aplikasi yang diwakili tombol "Quit". Interaksi dengan elemen tombol pada aplikasi sepenuhnya menggunakan *mouse*.

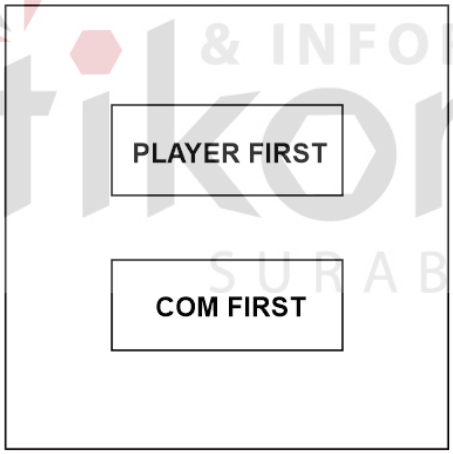
Gambar 3.3 Rancangan Tampilan Menu Utama

Bila pemain memilih untuk bermain, maka menu selanjutnya akan tampil. Pada Menu Tingkat Kesulitan, aplikasi akan meminta pemain untuk memilih tingkat kesulitan komputer. Tersedia tiga pilihan tingkat kesulitan, *Easy, Medium*, dan *Hard*.



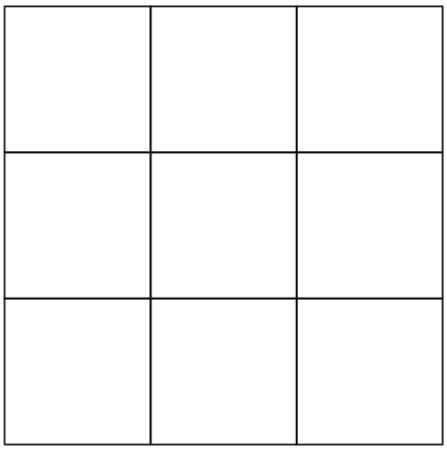
Gambar 3.4 Rancangan Tampilan Menu Tingkat Kesulitan

Setelah pemain memilih tingkat kesulitan, akan tampil pilihan apakah pemain ingin bermain dulu atau tidak. Bila pemain memilih tidak, maka komputer sebagai lawan main akan melangkah terlebih dahulu.



Gambar 3.5 Rancangan Tampilan Menu Pilih Sisi

Setelah pemain memilih, akan muncul papan permainan *Tic Tac Toe*. Pemain melangkah dengan klik kiri pada ruang kotak yang kosong, hingga tercapai kondisi GameOver (menang, seri, atau kalah).



Gambar 3.6 Rancangan Tampilan Papan Permainan

Setelah permainan selesai, akan muncul Tampilan Akhir Permainan, yang akan menunjukkan pemenang pada sesi permainan itu, serta akan menawarkan pilihan apakah peain ingin bermain lagi atau ingin keluar dari aplikasi. Gambar di bawah adalah contoh apabila pemain yang menang.

Gambar 3.7 Rancangan Tampilan Akhir Permainan

3.2.2 Perancangan Program

1. Membuat *header* SDL_Button.h

Tahap pertama pembuatan program adalah dengan membuat *header* (file berekstensi **.h**) yang berisi fungsi-fungsi dan telah dikompilasi sebelumnya. Apabila kita akan menggunakan suatu file header tertentu, maka kita akan mendaftarkannya melalui *directive* `#include`. Pustaka milik SDL seperti `SDL.h`, `SDL_ttf.h`, dan `SDL_image.h` disertakan pada *header* ini. Selain pustaka tersebut, dibuat juga *class* `SDL_Button` yang bertujuan sebagai bentuk penyederhanaan dari suatu permasalahan yang berkaitan dengan objek. Maka dari itu kelas didefinisikan sebagai sesuatu yang mempunyai data (sifat) dan fungsi (kelakuan).

Fungsi `SDL_Button` adalah fungsi inisialisasi untuk objek *button*. Fungsi-fungsi seperti `Button_SetWH`, `Button_SetPosition`, `Button_SetBackgroundColor`, `Button_SetText`, dan `Button_SetTextColor` memudahkan untuk melakukan aksi selama pembuatan program, karena *parameter-parameter* terkait dapat digabung menjadi satu menjadi sebuah fungsi.

2. Membuat *header* GUI.h

Tahap ini adalah tahap awal pembuatan antarmuka dengan menggunakan pustaka SDL. Pustaka milik SDL seperti `SDL_ttf.h`, dan `SDL_image.h` disertakan pada *header* ini. Selain pustaka tersebut, dibuat juga *class* `GUI` yang mempunyai tujuan utama mempersiapkan segala sesuatu yang berhubungan dengan

komponen antarmuka pustaka SDL yaitu `SDL_Surface`. `SDL_Surface` adalah struktur permukaan grafis yang nantinya akan digambar.

Fungsi yang ada pada *header* `GUI.h` di dalam *class* `GUI` adalah fungsi yang bertujuan mengatur proses urutan antarmuka menu program.

3. Membuat *header* Tictactoe.h

Tahap selanjutnya adalah pembuatan *header* untuk algoritma *Alpha-Beta Pruning*. Terdapat dua buah *struct*, yaitu *struct* `Board` untuk papan permainan dan *struct* `Move` untuk koordinat langkah. Selain *struct*, dibuat juga *class* `Tictactoe` untuk mendefinisikan properti-properti dari permainan *Tic Tac Toe* itu sendiri seperti anggota papan, posisi koordinat, dan kedalaman pencarian.

Fungsi yang ada pada `Tictactoe.h` di dalam *class* `Tictactoe` antara lain `initializeBoard` untuk menginisialisasi papan di awal permainan, `GetComputerMove` untuk mencari langkah komputer, dan `GetHumanMove` untuk langkah pemain manusia.

4. Membuat program utama main.cpp

Pada program utama main.cpp terdapat fungsi `main` yang berisi inisialisasi untuk pembuatan jendela program antarmuka. Selain itu juga ada pemanggilan `StateMachine` yang berfungsi untuk mengatur state keadaan permainan, apakah sedang berada pada menu awal, menu permainan (menu tingkat kesulitan, menu sisi, papan permainan), atau pada akhir permainan.

5. Membuat program utama SDL_Button.cpp

Pada program utama `SDL_Button.cpp`, fungsi `SDL_Button` akan mempersiapkan elemen tombol dengan properti-propertinya seperti warna tombol, posisi tombol di layar, dan jenis font. Bagian terpenting dari program utama `SDL_Button.cpp` adalah fungsi `Button_Events`. Fungsi ini bertugas untuk mendeteksi aksi dari *mouse*, yaitu posisi koordinat *mouse* dan klik kiri *mouse*.

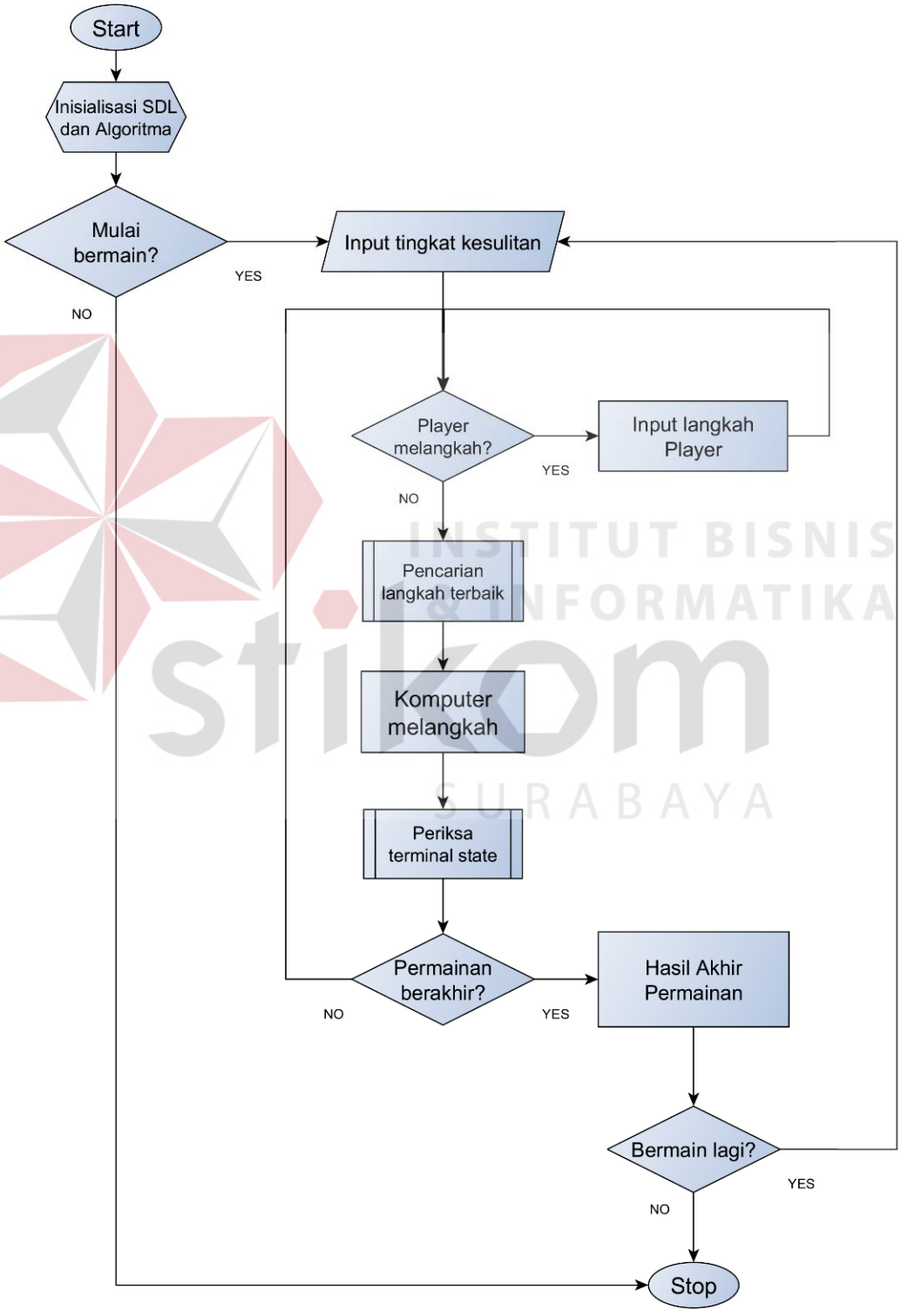
6. Membuat program utama GUI.cpp

Pada program utama `GUI.cpp`, fungsi GUI akan melakukan inisialisasi kanvas SDL dengan properti-propertinya seperti jenis font dan pemuatannya, warna font, pemuatan gambar *sprite*, serta pengaturan judul dari jendela (*Caption*). Fungsi-fungsi yang ada di `GUI.cpp` ini menentukan kanvas untuk masing-masing kondisi *state* permainan. Fungsi-fungsi tersebut antara lain `startScreen` (tampilan awal program), `menuScreen` (tampilan menu utama), `LevelSelectionScreen` (tampilan menu pemilihan tingkat kesulitan komputer), PlaySelection (tampilan menu pemilihan sisi), `RunGame` (tampilan papan permainan), dan `GameOverScreen` (tampilan ketika permainan berakhir).

7. Membuat program utama Tictactoe.cpp

Program utama `Tictactoe.cpp` berisi algoritma *Alpha-Beta Pruning*. Algoritma ini merupakan pengembangan dari algoritma sebelumnya yaitu algoritma *Minimax*. Penentuan langkah komputer, berdasarkan pada fungsi `minimaxAB` yang ada. Tingkat kesulitan komputer dapat disesuaikan dengan cara mengatur kedalaman pencarian oleh komputer. Nilai kedalaman pencarian mempunyai rentang antara 1 sampai 9, dimana semakin kecil nilai berarti tingkat

kesulitan semakin mudah, sementara semakin besar nilai berarti tingkat kesulitan semakin sulit. Adapun gambaran umum algoritma permainan *Tic Tac Toe* dengan *flowchart* adalah sebagai berikut.



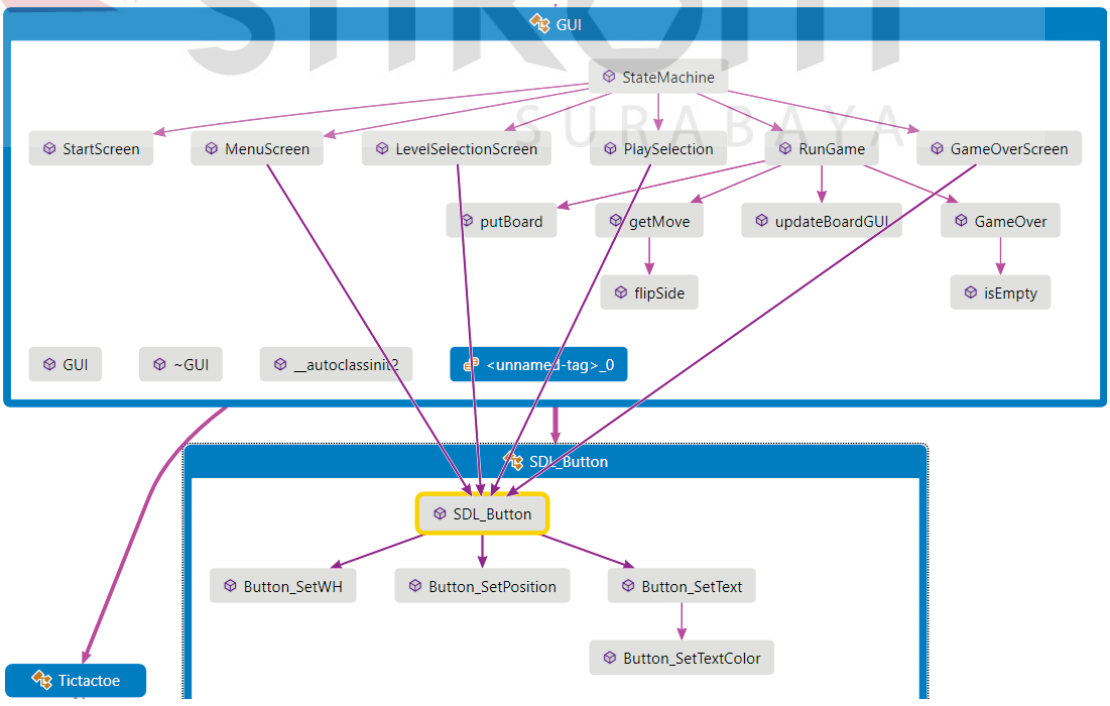
Gambar 3.8 *Flowchart* Program Permainan Tic Tac Toe

3.4 Implementasi SDL (*Simple DirectMedia Layer*)

Untuk menyelesaikan masalah pemrograman pada C++ yaitu dengan mendefinisikan *class-class* sebagai abstraksi dari objek-objek fisik. *Class* tersebut berisi keadaan *object*, anggota-anggotanya dan kemampuan dari *object*-nya.

Seperti yang telah dipaparkan pada subbab 3.2.1 tentang perancangan antarmuka menggunakan SDL, semua tampilan menu menggunakan SDL. Penyelesaian masalah GUI (*Graphical Unit Interface*) menggunakan *Class* `GUI`. Dari enam tampilan menu, empat diantaranya menggunakan elemen tombol sebagai kontrol navigasi. Keempat menu tersebut yaitu:

- Tampilan menu utama
- Tampilan menu tingkat kesulitan
- Tampilan menu pilih sisi, dan
- Tampilan menu akhir permainan

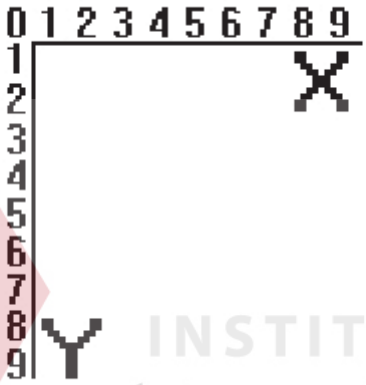


Gambar 3.9 *Code Map* Implementasi Menu dengan SDL

Untuk papan permainan sendiri, tidak menggunakan fungsi `SDL_Button` melainkan menggunakan *event handling*.

```
x = event.button.x;
y = event.button.y;
```

Event handling atau penanganan kejadian menggunakan `SDL_Event`. *Event handling* akan terus di-*polling* sampai terdeteksi aksi dari Pemain, dalam hal ini manusia. Sistem koordinaat SDL adalah seperti gambar di bawah.

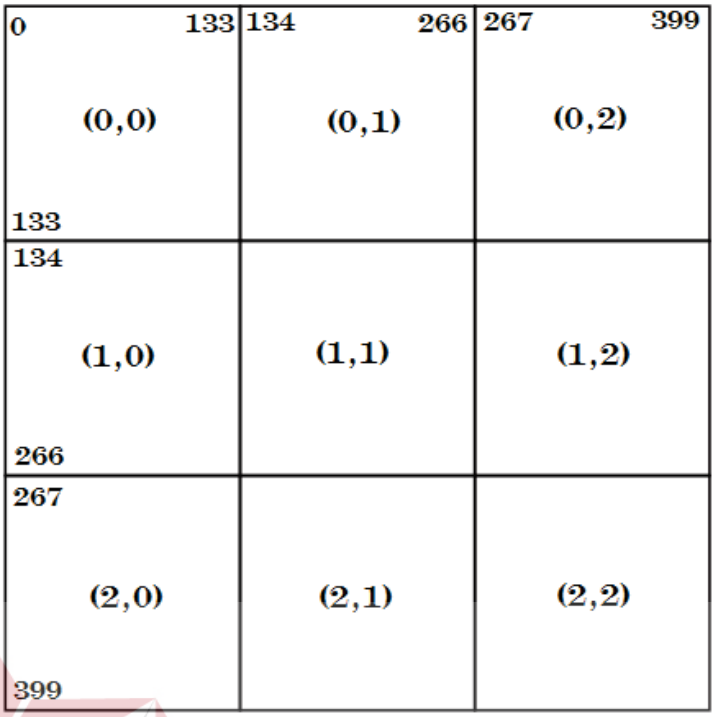


Gambar 3.10 Sistem Koordinat SDL
(Sumber: http://lazyfoo.net/SDL_tutorials/lesson02/index.php)

Posisi x dan y dari *mouse* saat melakukan aksi apapun, akan disimpan oleh `SDL_Event`.

```
if (game.getBoard().board[y / 133][x / 133] == 0)
{
        game.GetHumanMove(Side, game.getBoard(), y / 133, x / 133);
        flipSide();
}
```

Sehingga penentuan posisi langkah Pemain, dalam hal ini manusia, dapat diterjemahkan menjadi titik koordinat (baris, kolom), ditentukan oleh posisi piksel seperti gambar dibawah.



Gambar 3.11 Koordinat Papan Permainan

3.5 Implementasi Metode *Alpha-Beta Pruning*

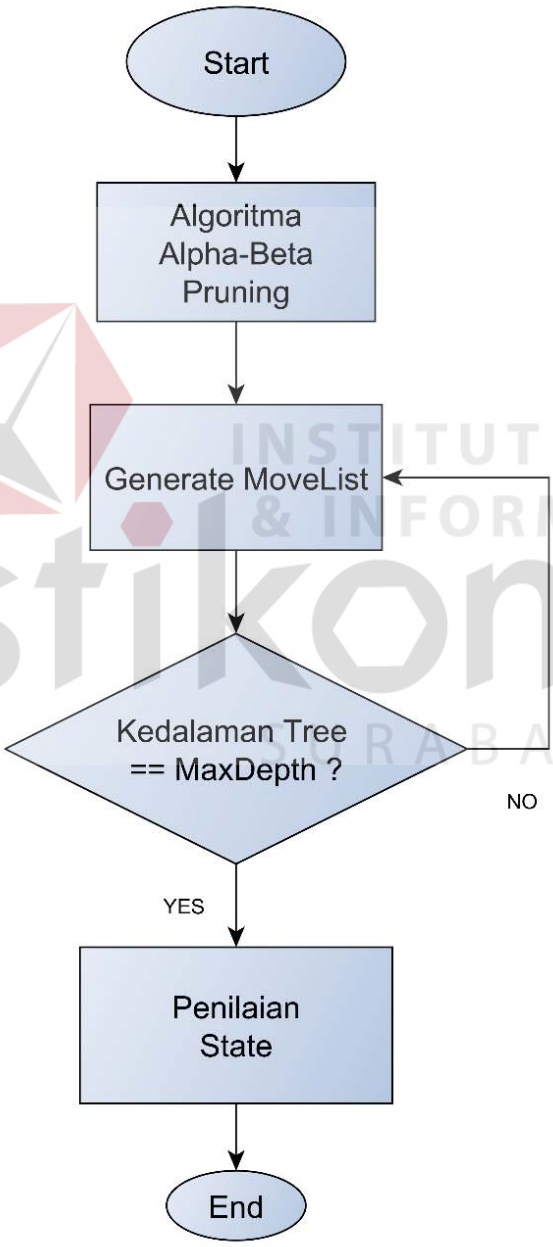
Secara teori, tingkat kesulitan dalam permainan yang menggunakan *game tree* dapat disesuaikan dengan cara mengatur kedalaman dari *game tree* yang dapat diproses oleh komputer. Semakin dangkal kedalaman yang dapat diproses komputer, maka dikatakan tingkat kesulitan adalah mudah. Sebaliknya, apabila semakin dalam maka dikatakan tingkat kesulitan adalah sulit.

```
if (easy.Button_OnClick())
{
        game.setMaxDepth(1);     ///Difficulty
        this->ScreenSelected = 3;
        SDL_WM_SetCaption("TIC TAC TOE - EASY", NULL);
}

if (medium.Button_OnClick())
{
        game.setMaxDepth(3);     ///Difficulty
        this->ScreenSelected = 3;
```

```
        SDL_WM_SetCaption("TIC TAC TOE - MEDIUM", NULL);
}

if (hard.Button_OnClick())
{
        game.setMaxDepth(5);     ///Difficulty
        this->ScreenSelected = 3;
        SDL_WM_SetCaption("TIC TAC TOE - HARD", NULL);
}
```



Gambar 3.12 *Flowchart* Pencarian Langkah Terbaik

Pertama-tama program akan menjalankan fungsi `RunGame`. Fungsi `RunGame` akan melihat pilihan sisi. Apabila pemain manusia melangkah terlebih dahulu, maka akan melakukan *polling* aksi dari *mouse* dengan fungsi `GetHumanMove`. Setelah didapatkan koordinat langkah pemain manusia, selanjutnya adalah giliran komputer. Fungsi `GetComputerMove` akan memanggil secara rekursif fungsi `minimaxAB` untuk mencari langkah terbaik komputer berdasarkan tingkat kesulitan atau nilai `MaxDepth` yang telah ditentukan.

```
void Tictactoe::GetComputerMove(int side, Board &board)
{
        this->bestMove.row = this->bestMove.col = -1;
        this->maxPly = 0;
        this->positions = 0;
        int alpha = -999999999;
        int beta = +999999999;
        minimaxAB(side, board, alpha, beta);
        board.board[bestMove.row][bestMove.col] = side;
        printBoard(board);
}
```

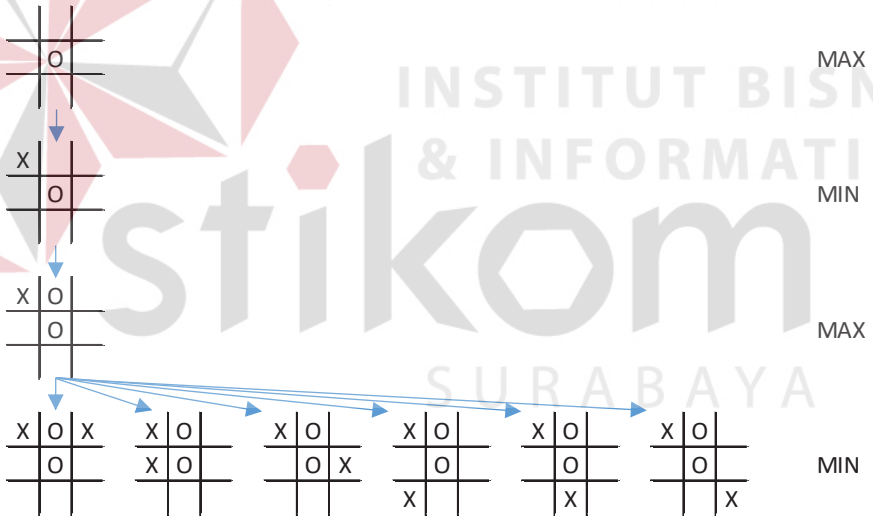
Pemilihan tingkat kesulitan (*easy*, *medium*, atau *hard*) akan mengubah nilai dari `MaxDepth`. Apabila `MaxDepth` bernilai 1, maka fungsi `minimaxAB` hanya akan melakukan rekursi sekali saja. Ini berarti komputer akan melihat dan mengambil keputusan permainan berdasarkan satu langkah kedepan. Apabila `MaxDepth` bernilai 3, maka fungsi `minimaxAB` akan melakukan rekursi sebanyak tiga kali. Ini berarti komputer akan lebih pandai mengambil keputusan karena komputer melihat tiga langkah kedepan. Demikian seterusnya.

```
move = moveList[index];
board.board[move.row][move.col] = side;   ///placing
printBoard(board);
ply++;
```

```
if (side == NOUGHTS)
        Score = minimaxAB(CROSSES, board, alpha, beta);
else
        Score = minimaxAB(NOUGHTS, board, alpha, beta);
board.board[move.row][move.col] = EMPTY;  ///remove placing
ply--;
```

Hal ini akan berpengaruh terhadap tingkat kesulitan dari komputer itu sendiri. Semakin jauh komputer melihat beberapa langkah kedepan, maka keputusan yang diambil komputer adalah keputusan yang makin menguntungkan komputer.

Fungsi `minimaxAB` akan menghasilkan *tree* sampai kedalaman `MaxDepth`. Dalam proses generasi untuk menghasilkan *tree*, dilakukan secara berurutan dan rekursif dari koordinat (0,0), (0,1) sampai (2,2).

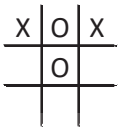


Gambar 3.13 Contoh *Tree* Kedalaman 3

Setelah itu dilakukan penilaian untuk keadaan papan tersebut dengan fungsi `Eval`. Fungsi `Eval` akan menghitung nilai keadaan papan menggunakan fungsi `PossibleWinningLines` untuk tiap sisi/pemain.

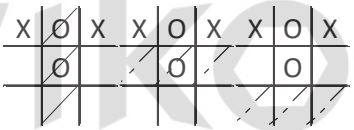
```
int Tictactoe::Eval(int side, Board &board)
{
    if (side == CROSSES)
    {
        int Not_side = NOUGHTS;
        return (PossibleWinningLines(board, side) - PossibleWinningLines(board, Not_side));
    }
    else
    {
        int Not_side = CROSSES;
        return (PossibleWinningLines(board, Not_side) - PossibleWinningLines(board, side));
    }
}
```

Perhatikan contoh keadaan papan permainan di bawah.



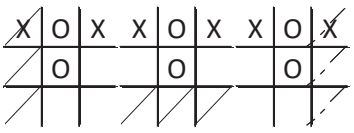
Gambar 3.14 Salah Satu Contoh Keadaan Papan

Untuk sisi O ada kemungkinan mendapatkan 3 simbol segaris sebanyak 3 kemungkinan.



Gambar 3.15 Kemungkinan 3 Simbol Segaris Sisi O

Untuk sisi X ada kemungkinan mendapatkan 3 simbol segaris sebanyak 3 kemungkinan.



Gambar 3.16 Kemungkinan 3 Simbol Segaris Sisi X

Maka, untuk keadaan demikian nilai keadaan papan tersebut adalah

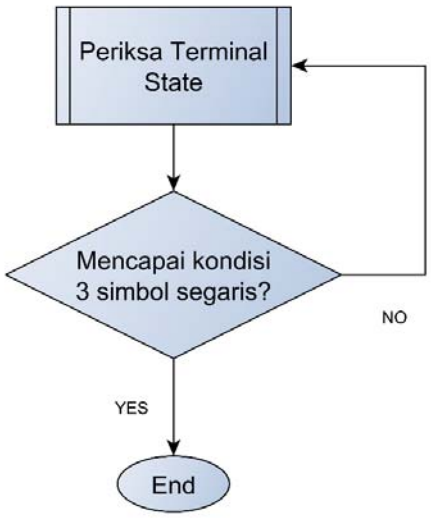
fungsi `PWL (O)` – fungsi `PWL (X)` = 3 – 3 = 0

Setelah kedalaman sampai dengan `MaxDepth`, maka fungsi akan menghitung nilai α dan β.

```
if (Score > bestScore && side == CROSSES)
{
        alpha = bestScore = Score;
        bestMove = move;
}
else if (Score < bestScore && side == NOUGHTS)
{
        beta = bestScore = Score;
        bestMove = move;
}
```

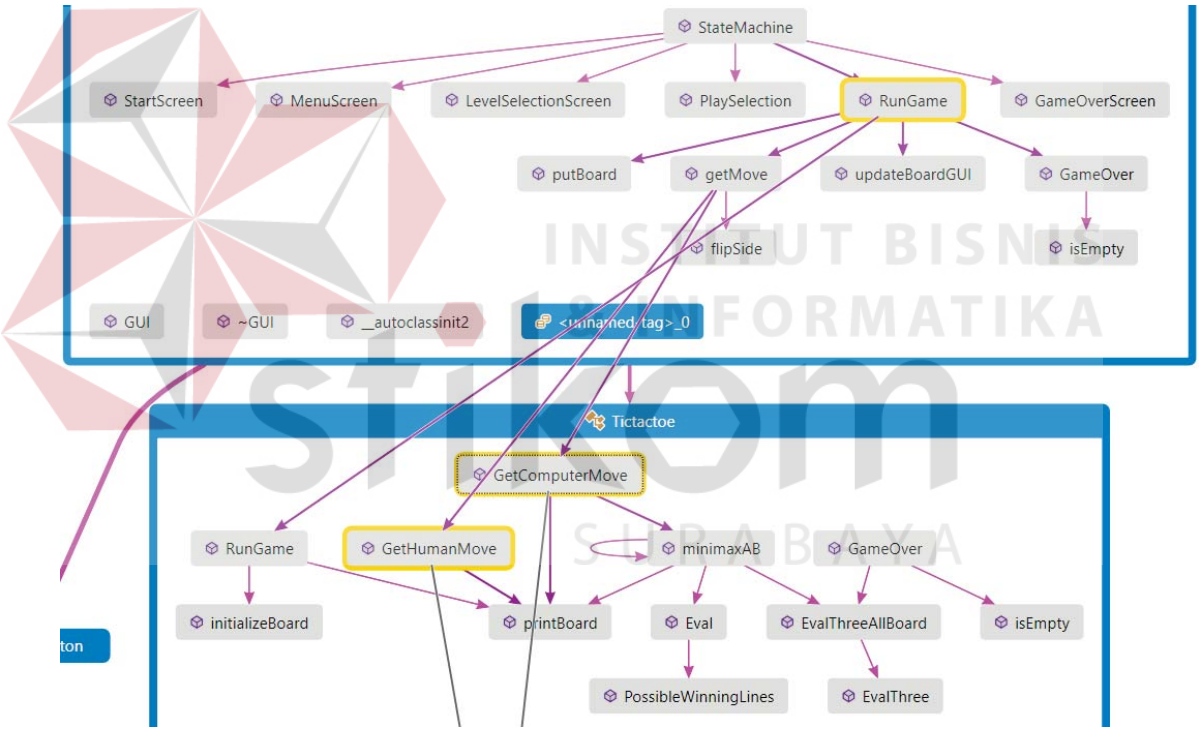
Pruning (pemangkasan) terjadi ketika nilai α lebih besar dari β. Saat itu terjadi, fungsi akan mengembalikan nilai `bestScore` (nilai terbaik).

```
if (alpha >= beta)
{
        return bestScore;
}
```



Gambar 3.17 *Flowchart Terminal State*

Untuk menentukan hasil akhir dari permainan (menang maupun seri) fungsi `EvalThree` dan `EvalThreeAllBoard` akan dipanggil tiap rekursi `minimaxAB`. Fungsi `EvalThree` dan `EvalThreeAllBoard` akan memeriksa apakah dari sisi O maupun sisi X ada yang sudah mencapai tiga simbol segaris. Bila salah satu sisi telah mencapai kondisi tersebut, maka pemain di sisi itu dinyatakan sebagai pemenang. Bila kedua sisi tidak ada yang mencapainya, maka permainan dinyatakan seri.



Gambar 3.18 *Code Map* Implementasi Algoritma Permainan

3.6 Metode Analisis

Metodologi analisis yang digunakan dalam penelitian Tugas Akhir ini adalah dengan menganalisis hasil akhir aplikasi, bagaimana kesesuaian algoritma *Alpha-Beta Pruning* yang diaplikasikan pada permainan *Tic Tac Toe* serta

bagaimana implementasi SDL (*Simple DirectMedia Layer*) sebagai antarmuka aplikasi dengan program. Hasil akhir tersebut berdasarkan pada dua pengujian, yaitu:

1. Pengujian SDL (*Simple DirectMedia Layer*) sebagai antarmuka aplikasi dengan program
2. Pengujian algoritma *Alpha-Beta Pruning* sebagai kecerdasan komputer