FPSでのリアルな思考を持つAIの試作

谷聖一 研究室　西田 智博
Tomohiro Nishida

概要

1970 年代に FPS(First Person Shooting) というゲームのジャンルが確立した．それ以来，様々な FPS の作品が世に出た．近年，FPS の中でもプレイヤーが操作しないキャラクターに AI(Artificial Intelligence) を搭載して，敵・味方として戦うことがメインの作品も増えてきた．報告者が知る限り，一週間程度で新しい武器・ステージを実装した環境に適応する少人数用のオンライン FPS の AI はない．そこで，新しい武器・ステージを実装した環境に適応する AI をプレイヤーの動作を利用して試作した．

1　はじめに

FPS とは一人称視点のシューティングという意味があり，まるで自分がその場に立っている・操っている感覚を覚えるゲームである．基本的には，プレイヤーが銃の様な物を撃ったりするのが定番で，ネット上で戦う FPS のことをオンライン FPS と呼ぶこともある．例えば，「Alliance of Valiant Arms」([1]) がある．

1.1　どのような FPS を想定しているのか

今回はオンラインゲームで現代武器を取り扱う様な FPS を想定して考えている．なぜなら，オンラインゲームはゲームバランスの変化が激しいため，プレイヤーの立ち回りが変化しやすいのが理由である．

想定しているゲームルールは以下の通りである．

- 8 対 8 の計 16 人
- 復活なし
- 1 試合は 3 分
- 敵を全滅させることが目標
- 目標を達成したチームが勝利する (時間切れの場合，生き残りが多い方が勝利)

復活なしとした理由は，立ち回りをうまく扱いたいので，緊張も少ない復活ありでは期待しているログのサンプルが取れないと考えた．

1.2　なぜ新しい AI が必要か

Artificial Intelligence(人工知能) は，太原によると「『考える機械』を作ることを目的とした研究領域」([2]) である．また本演習では，NPC(None Player Character) に搭載する人工知能のことを AI と呼ぶ．

報告者はオンラインゲームのゲームバランスの変化が特に激しい環境で，一週間程度の短い期間で AI が適応するオンラインゲームの AI モデルは見たことがない．従来は，元々の NPC の設計が良かったり，新しい武器・ステージ等が実装されることを想定して，AI の難易度を調節することで上手く適応していた．

AI はいくらでも強くすることができて，人間はほぼ勝てない．例えば，足音でプレイヤーの位置を正確に把握し，一瞬で全体を見渡せる正確な視点操作ができ，壁の貫通を考えても当てられる位置に移動してくる敵にヘッドショット (頭に当てること) により，ほんの 1・2 発でその敵を倒せる．しかし，そのような強すぎる AI では直ぐにユーザーは離れてしまう．逆に，設定を甘くすればいいという訳でもない．設定を甘くしすぎればプレイヤーは敵を無双できるが，回数を重ねるごとに単調な作業を続けている様な疲れがプレイヤーを襲う．今日のオンライン FPS では，元々設定された各難易度をプレイヤーに任意で選んでもらう形で，自らが欲しい刺激を選んでいる．今までの AI のモデルでは適度にプレイヤーを刺激する様な AI を作ることは難しいと思ったので，新しい AI のモデルを作ることが必要だと考えた．

1.3　激しい立ち回りの変わり目

オンライン FPS の特に無料オンライン系 FPS は新しく強い武器を出す傾向が強く，プレイヤーの動きも自ずと変わっていく．大体は数戦程戦えば，自分と新武器を持った相手とどちらの武器が優っているかが概ね分かる．そうなると，自ずとより有利になりやすい立ち位置を保とうと詮索・(位置の) 保持等で立ち回りに変化が出て来る．そして，数日経てば有利な武器を持ち・勝てるポジションを自分なりに持ち，一部の場所に止まることが多くなる．これが立ち回りの「大きな変わり目」で，武器のバランス調整も同様だと考えている．また，プレイヤー自身のスキル向上やその逆も考えると立ち回りの変化はプレイヤーの数だけ刻々と変わっていく．

2　どういう AI を目指すのか

目標は新しい環境になった際，短い時間でその環境に適応した戦略を持つ AI のモデルになること．より具体的な AI のモデルは，過去のプレイログと照らし合わせて適合すれば使用し，利用するプレイログについては全部のログをシュミレーションしてもらいスコアの高いものから一定のリストを使用する．

3　3D ゲームの作成

3D ゲームを作成する為に 3D モデルを使ったゲームを 1 から学ぶ必要があると考えた．よって，DirectX9 と Microsoft Visual Studio C++でこのモデルの基礎となるプログラムを作り，Unreal Engine 4(C++および BluePrint) で，Unreal Engine 4 の強力な技術を利用しながらよりグラフィックの良いゲームとして AI のモデルを落とし込むことにした．

XSI Mod Tool という無料で 3D モデルを作成できるソフトを用いて，テキストで内容を確認できる X ファイルで，図 1 図 4 のように武器・キャラクター・ステージ・弾を作成した．

図 1: 作成したスナイパーライフルの 3D モデル

図 2: 作成したアサルトライフルの 3D モデル

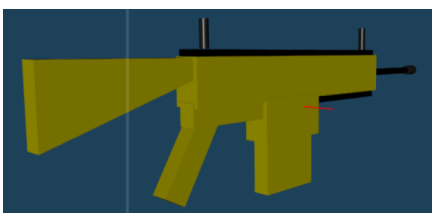

図 3: 作成したアサルトカービンライフルの 3D モデル

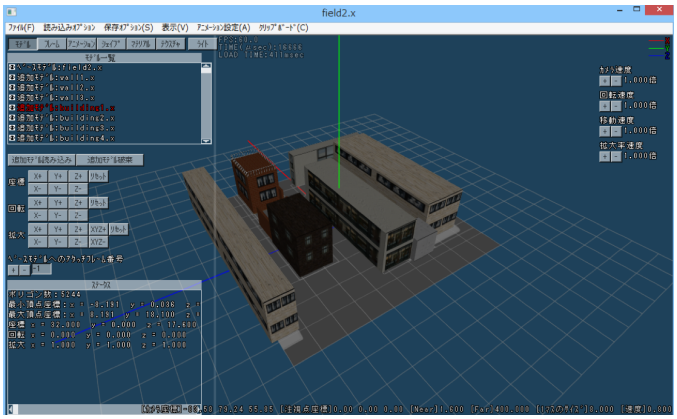

図 4: 作成したステージ

4　AI の試作

4.1　簡単な AI 作成

以下の 6 段階のステップで NPC が移動できる線を作ってその上を移動し，攻撃する．

1. ステージの中央からキャラクターが移動できる程の幅をもつ長方形を探して作る．(図 5 図 7 参照)

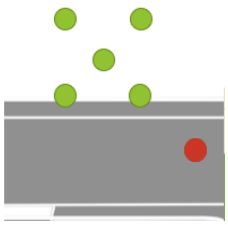

図 5: 中央からキャラクターが移動できる程の幅をもつ場所を探す

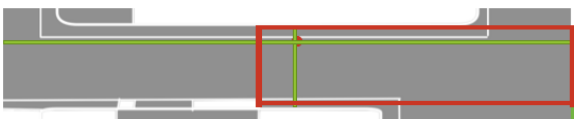

図 6: 見つけたら縦横の線を伸ばして，短い方を縦横の移動できる方に左右移動する

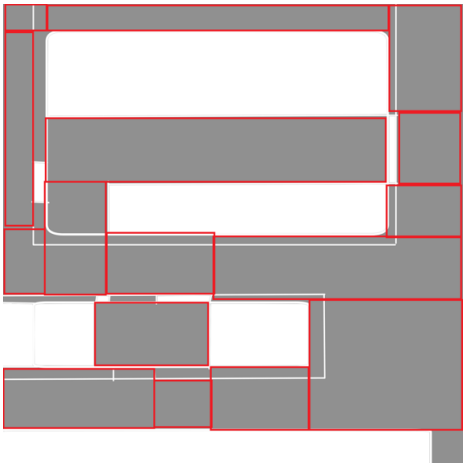

図 7: 可能な限り長方形を作成した

2. 先ほどの長方形を中央から縦・横の中央に線を伸ばし、隣の長方形に繋がる様に線を伸ばす．(図 8 図 11 参照)

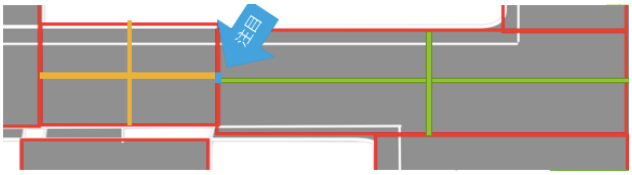

図 8: 上下左右、縦横のどちらかの座標が重なり・障害物がないものを繋ぐ

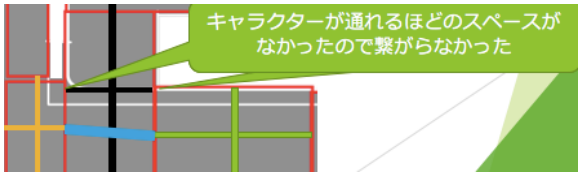

図 9: 上下左右繋がっていない線で、他の長方形の逆方向の線が繋がってなく、繋ぐ線に障害物がないものを繋ぐ

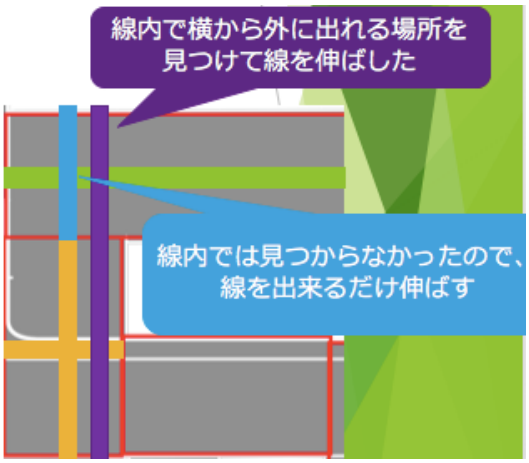

図 10: 上下左右繋がっていない線で、注目している長方形以外の通れる道に線を伸ばす

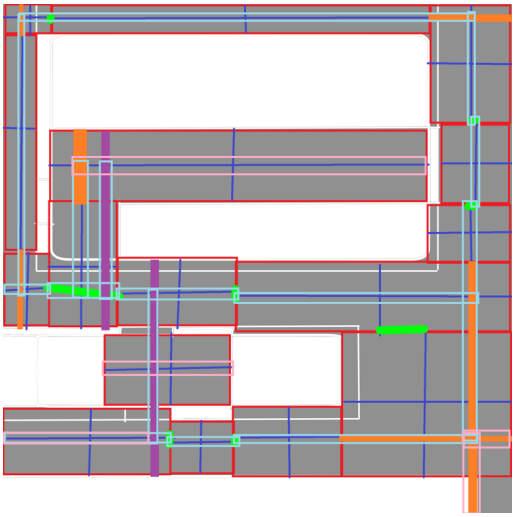

図 11: 可能な限り繋げ・伸ばした線

3. 線で繋がった長方形同士・道のり・距離のリストを作る．(図 12 参照)

```
[DebugPrint]: i:0 j:10 dict0:395.847 Line0:18
[DebugPrint]: i:0 j:10 dict0:395.847 Line0:7.40808
[DebugPrint]: i:0 j:10 dict0:395.847 Line0:33.7039
[DebugPrint]: i:0 j:10 dict0:395.847 Line0:7.40808
sWrong?:0
[DebugPrint]: i:0 j:10 dict1:395.847 Line1:33.7039
[DebugPrint]: i:0 j:10 dict1:395.847 Line1:7.40808
[DebugPrint]: i:0 j:10 dict1:395.847 Line1:33.7039
[DebugPrint]: i:0 j:10 dict1:395.847 Line1:2
sWrong?:0
[DebugPrint]: i:0 j:10 dict2:395.847 Line2:33.7039
[DebugPrint]: i:0 j:10 dict2:395.847 Line2:2
[DebugPrint]: i:0 j:10 dict2:395.847 Line2:33.7039
[DebugPrint]: i:0 j:10 dict2:395.847 Line2:-3.5
sWrong?:0
[DebugPrint]: i:0 j:4 dict0:512.865 Line0:18
[DebugPrint]: i:0 j:4 dict0:512.865 Line0:7.40808
[DebugPrint]: i:0 j:4 dict0:512.865 Line0:33.7039
[DebugPrint]: i:0 j:4 dict0:512.865 Line0:7.40808
```

図 12: 作成したリストの一部

4. 先ほどのリストは隣接しているものとして扱い，ワーシャル・フロイド法を用いて最小の距離かつ全ての開始地点から到着地点を出す．(図 13 参照)
ワーシャル・フロイド法は，隣接する全ての頂点から全ての頂点への経路の長さを見ていき，最短経路を見つければ，更新するアルゴリズム ([3]).

```
i:0 j:1 dict:513.313 processNum:1 process:1
i:0 j:2 dict:1349.63 processNum:4 process:1
i:0 j:2 dict:1349.63 processNum:4 process:12
i:0 j:2 dict:1349.63 processNum:4 process:5
i:0 j:2 dict:1349.63 processNum:4 process:2
i:0 j:3 dict:1139.49 processNum:5 process:4
i:0 j:3 dict:1139.49 processNum:5 process:9
i:0 j:3 dict:1139.49 processNum:5 process:8
i:0 j:3 dict:1139.49 processNum:5 process:6
i:0 j:3 dict:1139.49 processNum:5 process:3
i:0 j:4 dict:330.766 processNum:1 process:4
i:0 j:5 dict:1101.12 processNum:3 process:1
i:0 j:5 dict:1101.12 processNum:3 process:12
i:0 j:5 dict:1101.12 processNum:3 process:5
i:0 j:6 dict:935.295 processNum:4 process:4
i:0 j:6 dict:935.295 processNum:4 process:9
i:0 j:6 dict:935.295 processNum:4 process:8
i:0 j:6 dict:935.295 processNum:4 process:6
i:0 j:7 dict:1526.74 processNum:4 process:4
i:0 j:7 dict:1526.74 processNum:4 process:10
i:0 j:7 dict:1526.74 processNum:4 process:11
i:0 j:7 dict:1526.74 processNum:4 process:7
```

図 13: ワーシャル・フロイド法を用いて作成したリスト

5. 線の上を歩く．先ほどのリストを用いて目的地まで移動する．

6. 指定した視界に入ったら敵の方向を向いて攻撃する．(図 14 参照)

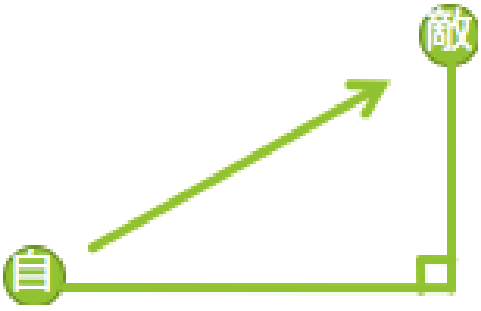

図 14: 自分と敵の位置の差を取って目標への向きを作る

4.2 ログを使う AI 作成

ログを使う前にログを出力しなければならない．今回は，移動しているか・射撃しているか・Shift を押しているか (音を出さずに歩く)・Jump しているか・長方形の番号・場所・移動量・向いてる方向の要素をファイルに出力した．次に，6 段階のステップで優秀なスコアを出したログを使う．

1. 利用できるログのリストを作成する．(図 15 参照)

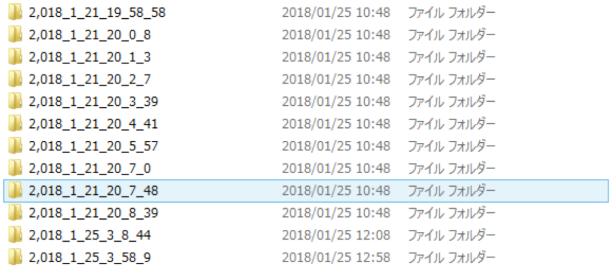

図 15: フォルダ直下のファイルを読み込む

2. それぞれのログの要素を読み込む．(図 16 参照)

図 16: インデックス番号 0 番目のログの中身

3. ログを利用して、シュミレーションする．(図 17 と図 18 参照)

図 17: 固定せずに可能な限り処理する

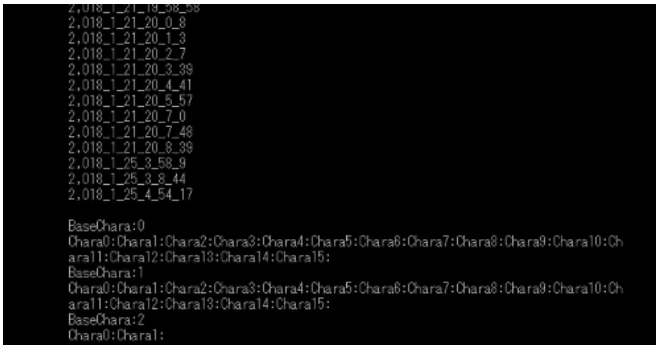

図 18: フォルダ名から直下のファイルをロード

4. 優秀なスコア順からフォルダ名をファイル出力する．(図 19 参照)

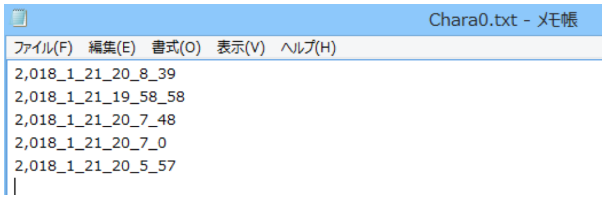

図 19: フォルダ名をファイルにそれぞれ書き出す

5. 優秀なフォルダのファイルを読み込む．(図 20 参照)

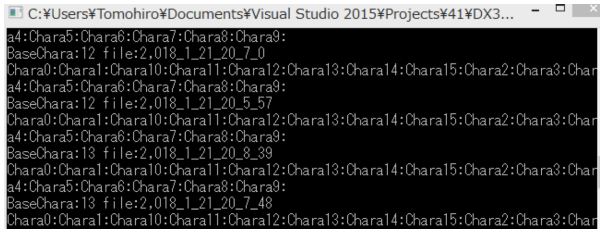

図 20: 読み込んだファイルをコンソールに出力

6. 自分の状況によって使用するログを決める．
   注目するキャラクターのインデックス番号のみで，同じ長方形がいる．次に，先ほどのキャラクター以外で，同じ長方形にいる．かつ，お互いの距離がより近いものが少なくともあれば，選ばれたログを使用する．また，この処理は対他的にしているので，該当しなければ今設定しているログか簡単な AI を使用する．

4.3 UnrealEngine4 に落とし込む

3 段階のステップで、若干変えて AI を Unreal Engine 4 に入れる

1. マップ作成 (図 21 参照)

図 21: 作成したマップ

2. マップに長方形を作成 (図 22 参照)

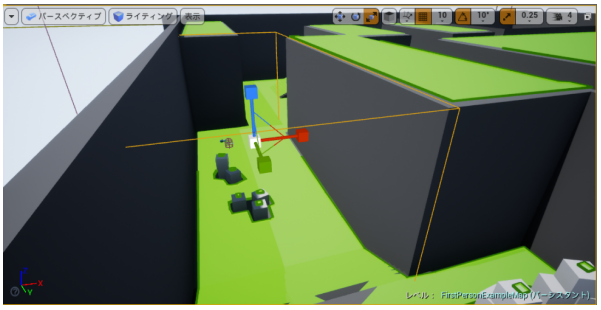

図 22: 手動で長方形を作成

3. AI を入れる (途中)(図 23 参照)

図 23: プレイ中のスクリーンショット

5 結果と考察

5.1 試作した AI のモデルの有効性

本演習で試作した AI モデルが有効かについては，プレイヤーの動きに沿った立ち回りになること以上は分からない．何故なら，本格的に十分な量のログが取れない原因が 3 つあった．

1. 完成が遅かったので高速に読み込む技術を組み込めなかった．
2. 最初は合計約 160 戦からプレイヤーだけを取ってきているので実際は 10 戦になっていた．
3. ファイル読み込みが長くログの数が少なかった．(状況を読む AI では数十分くらい待つ)

しかし，十分な量のログを使った試合はできず，少量のログではあったが，ログを使う AI を使うことで簡単な AI との試合と比べて 3 つの変化を確認できた．

1. 簡単な AI では開始直後に、赤チームの NPC が纏まって青チームを一掃することがたびたびあったが，ログを使う AI では一方的な試合は少なくなった．
2. 開始から大きくまとまることが少なくなった．

3. 敵の動きが読みにくかった．

これらは報告者のプレイスタイルであり，実際に同様の現象が必ず起こる確証はない．

5.2 自作 AI の課題

今後の課題として次の三つを挙げる．

1. 試合時間が長くなるほど簡単な AI のプレイが長くなるので，利用できるログが少なくなる．
2. チームで固まった動き (または放置) の方が，極度に強い可能性がある．
3. 数人でプレイする際の AI の強さ調整は，スコア/デスの割合に近いログを使用すれば上手くいく可能性がある．

参考文献

[1] Alliance of Valiant Arms 公式ホームページ https://ava.pmang.jp/

[2] 太原 育夫 新 人工知能の基礎知識 (近代科学社) 2008/6/30 発行

[3] ワーシャル・フロイド法 https://jaΓwikipediaΓorg/wiki/%E3%83%AF%E3%83%BC%E3%82%B7%E3%83%A3%E3%83%AB%E2%80%93%E3%83%95%E3%83%AD%E3%82%A4%E3%83%89%E6%B3%95(2017/9/1 参照)