AI Term Project Report Group 1

B94902022 周尙柏 B94902062 江愷陽
B94902063 呂鈺達 B94902065 余守壹

- **Problem Definition:**

我們這組主要的目標是創造出一個可以讓人和 AI 互相對抗的賽車環境，因此我們主要把 problem definition 分成三大部分。

1. 讓 AI 控制其中一台賽車，使其能順利且快速的完成整個賽程。
2. 建構出整個比賽的環境(包括賽道，起跑線，倒數計時等功能)。
3. 讓人類玩家可以用鍵盤來操控另一台賽車與 AI 控制的賽車競賽。

- **Proposed Solution -- Emphasizing AI Techniques:**

關於第一個功能，也就是讓 AI 可以順利完成賽程的部分，我們先用 A-star Algorithm 建出整張地圖的 potential field，終點的 potential 爲最低，藉此導引賽車往終點跑。A-star 所算出來的 potential 不受到牆壁影響，所以做完 A-star 以後，會再算牆壁所給的 potential，越接近牆壁會有越大的排斥力把機器人推離牆壁。這樣子機器人就可以透過已經建好的 potential field 來知道要往哪裡跑。接下來是修改 MRS 中機器人底層 Drive 部分的 Code，讓機器人可以順利過彎。第二個功能我們主要是修改 Maze simulator 的部份，加入起跑線，並且設定兩輛車，最後再加上起跑前倒數計時的功能。如何讓人控制機器人則是修改原本 MRS 所提供 Dashboard 的 code，將鍵盤與 Dashboard 連結起來，達成可以透過鍵盤操控賽車的目的。

- **Contributions (highlight any special features/findings):**

  - **建立 Potential Field**

主要是修改 ExplorerSim.cs。我們先把地圖切細，map 中的每一個 pixel 我們又把它切成三倍細，這樣子算 potential 時比較精確。Potential field 是用一個二維陣列來記錄，在地圖上每一個空白的點都會有一個(a_x, a_y)。表示說機器人在那一個點時，會受到那麼大的加速度。

1. A-star Algorithm

是直接用 Task 2 寫的 A-star。原本的 heuristic 有考慮 Manhattan Distance，但後來想說其實沒有差很多，所以就刪掉了。最後用的 heuristic 就是轉彎次數越少的 path 優先。我們的 A-star 可以有 8 個方向，即原本的四個方向加上斜向的四個方向。不過多加了斜向後，會讓機器人貼牆貼的更近，每次轉彎時都會被牆壁排開，反而轉得更不順。

2. 牆壁的排斥力

牆壁的力和其距離成反比。所以在牆壁附近(程式是設 3 格)，就會受到一個斥力，距離牆壁越遠，斥力越小。

■ AI 賽車控制

1. 邊轉彎邊開車

更改 SimDiffDriveEntities.cs 裡面的程式碼。其實逼迫車子一定要轉彎才能停的地方只有一個，在 *Update()*函示裡面：

```
// This is cheating! Eliminates inertia

_leftWheel.Wheel.AxleSpeed = 0;
_rightWheel.Wheel.AxleSpeed = 0;
```

這段程式碼會逼迫車子在轉彎完後，把左右輪的速度都設爲零，而且不考慮慣性！控制速度考慮慣性的方法是用 *SetAxleVelocity()*。這樣設計使得車子在轉動的時候，一到想要的角度，就會馬上停下不再轉動。但如果考慮慣性的話，車子還需要花一點時間把速度降下來，因此車子就會轉過頭。爲了避免轉過頭，我們就寫當剩下的角度越來越少時，轉動的 power 就越調越小，這樣即使轉過頭，但因爲轉動的速度已經很慢了，所以不會轉過頭太多。程式現在就可以邊呼叫 *MoveForward()*，邊呼叫 *Turn()*。不再有車子停下來才能轉的限制。

在 ExplorerSim.cs 中呼叫 *MoveForward()*會跑到 SimDiffDriveEntities.cs 的 *SetVelocity()*。*Turn()*會跑到 *StartTurn()*。*Translate()*會跑到 *StartTranslate()*。而我們有修改上面有提到 SimDiffDriveEntities.cs 中的三個函示，改成設定速度/要轉的角度的功能。

2. Potential Field 如何影響車子速度

我們車子原本設計是，車子在哪一格，就讀那一格的加速度爲何，然後把它加到現在車子的速度上面。得到了新的速度以後，就要和原本的速度去比較，算出需要轉多少角度，然後呼叫 *Turn()*。算速度也一樣，算出現在速度向量的長度爲何，然後呼叫 *MoveForward()*。我們也有限制車子的最高速，若車子的速度向量超過了最上限，X Y 就會被等比例的壓下來。是控制速度和轉彎的兩個函示。

3. 車子會 lag?!反應很慢

前面有提到，車子在哪一格，就會讀那一格的加速度。但是因爲車子下指令和真的執行之間會有一段蠻長的時間差。因此車子開始轉彎時，早就經過該轉彎的地方了。因此，我們就加入了一個 look-ahead vector，由現在的速度向量算出來的。就是會以車子行經方向前約零到三格的加速度當做加速度。若車子速度越快，會看的越遠，速度越慢，就看的越近，甚至看他現在在的那一格。這樣子，車子的反應就不

會再那麼慢了。

■ 比賽環境

1. 起跑線與賽車繞圈

我們在地圖上多加了起跑線，以及起跑線的參考原點和終點。起跑線在一開始在 path planning 和建 potential 時會先視爲牆，利用起跑線兩旁的參考原點和終點用 A star 算出一條 path。然而，雖然大部分的 path 和 potential 都是正確的，但起跑線附近的 grids 卻因爲起跑線在一開始被設爲牆壁而被排開，若要做到賽車繞圈，我們必須讓賽車能通過起跑線(而不是被起跑線排斥)。於是我們在 path planning 和 potential 建好後，會針對起跑線旁的這些 grids 修正其 potential，讓車子到達參考終點後能順利的通過起跑線再繼續繞圈。

2. 倒數計時

修改 MazeSimulator.cs 裡面的 *WatchDogUpdateHandler()*，程式開始前，Differential Drive 會被鎖住，直到開始爲止，才會被解開，車子才會動。

■ 人操控賽車

我們主要是修改 DriveControl.cs 這個檔案，多用了 KeyDown，KeyUp 和 KeyPress 這幾個 handler 來抓鍵盤的資訊，抓到鍵盤資訊以後再去分別呼叫 *OnTranslate()*來前進或是後退，以及呼叫 *OnRotate()*來轉向，而且我們用的方法可以同時抓到兩個以上的按鍵的資訊，使得可以達成邊按前進邊轉向的功能。

● Demo/Experiment Result:

路寬度只有一格的時候，車子會因爲反應不夠靈敏，一直開不進去。所以車子路的寬度至少要兩格。若路都是兩格的話，在 219 的電腦上跑，車子可以開的很順，也不會有什麼危險鏡頭。但是一換到筆電上面去時，連兩格的路也開不過去。我們發現車子的反應會因電腦而異，所以換一台電腦跑時，就要重新調整一些參數。還頗麻煩的。

我們最後做出來是能夠順利過彎，有障礙物時也會閃躲。但是因爲沒辦法在文字上呈現，因此就錄了影片，也順便當做紀念。影片在 demo 的時候也有放出來。

● Job Responsibility of each Member

1. B94902022 周尚柏：人控制賽車、起跑線、投影片、上台報告。
2. B94902062 江愷陽：Potential Field建造、賽車繞圈。
3. B94902063 呂鈺達：人控制賽車、賽車轉彎角度計算、賽車繞圈、錄製影片。
4. B94902065 余守壹：賽車邊跑邊轉、賽車參數微調、錄製影片、上台報告。
5. 報告大家都有寫。