TRƯỜNG ĐẠI HỌC CÔNG NGHỆ
ĐẠI HỌC QUỐC GIA HÀ NỘI

BÁO CÁO BÀI TẬP LỚN
Môn học: Các vấn đề hiện đại trong công nghệ thông tin

DỰ ÁN GAME CỜ TƯỚNG VR HỖ TRỢ NHẬN DIỆN GIỌNG NÓI

Giảng viên: PGS Lê Sỹ Vinh

Sinh viên: Đào Quang Hiếu - 19020289
Đỗ Hồng Hà - 19020076
Bùi Chí Trung - 19020054

Tháng 12, năm 2022

Phân công công việc

Công việc chung	
Học khoá học Unity Essentials [7] và Junior Programmer [4] để biết cách sử dụng Unity.	
Xây dựng chuẩn cờ tướng.	
Thiết kế logic game.	
Debug và test game.	
Công việc của từng thành viên	
Đào Quang Hiếu (40%)	Học lý thuyết VR.
	Xây dựng, cài đặt logic VR.
	Xây dựng, cài đặt thành phần Command Parser.
	Xử lý phát hiện giọng nói (voice activity detection).
	Xây dựng API cho nhóm AI sử dụng.
	Xây dựng model cho môi trường trong game.
Bùi Chí Trung (30%)	Xây dựng kiến trúc game.
	Cài đặt thành phần Game Logic.
	Thiết kế UI trong game.
	Thu âm và xây dựng Sound Effect, Particle Effect.
	Tối ưu hóa game.
Đỗ Hồng Hà (30%)	Học lý thuyết VR.
	Xây dựng, cài đặt logic VR.
	Xây dựng model cho bàn cờ và quân cờ.
	Cài đặt thành phần Game Logic.
	Build game và container hoá trên docker.

Mục lục

Danh sách hình vẽ

1 Chủ đề tìm hiểu: VR - Thực tế ảo

Thực tế ảo là một trong những chủ đề được mọi người, kể cả các tập đoàn công nghệ lớn quan tâm trong thời gian gần đây.

Từ những trải nghiệm trong môi trường thực tế ảo, nhóm chúng em muốn tạo ra một sản phẩm có thể thể hiện được những thế mạnh của việc trải nghiệm thực tế ảo so với trải nghiệm 3D qua màn hình theo cách truyền thống.

1.1 Phân biệt một số khái niệm liên quan đến VR

Dưới đây là một số khái niệm mà khi mới bắt đầu tìm hiểu về VR, nhiều người có thể nhầm lẫn:

- VR - thực tế ảo: công nghệ sử dụng các thiết bị như kính VR, tay cầm để tạo ra những âm thanh, hình ảnh ảo khiến người dùng cảm giác như đang như đang ở trong một thế giới khác.

Hình 1.1: Thế giới ảo trong VR.

- AR - thực tế tăng cường: công nghệ cho phép lồng ghép thông tin ảo vào thế giới thực. Công nghệ này hiện nay được dùng chủ yếu trên điện thoại, máy tính bảng. Ví dụ: game Pokemon Go(khi mở camera máy, ngoài cảnh vật bình thường xung quanh, trên màn hình còn có thể xuất hiện con vật ảo hay còn gọi là pokemon).

Hình 1.2: Game Pokemon Go

- MR: là sự kết hợp giữa công nghệ VR và AR kể trên:
  - Yếu tố AR: cho phép lồng ghép các đối tượng ảo vào thế giới thực.
  - Yếu tố VR: cho phép tương tác được với các đối tượng ảo được hiển thị kể trên.
- XR: là khái niệm bao trùm cho tất cả các thuật ngữ VR, AR, MR.

1.2 Ứng dụng của VR

Ngoài ứng dụng trong lĩnh vực giải trí như làm game, phim ảnh nhiều người vẫn tưởng. VR có thể được ứng dụng trong rất nhiều lĩnh vực đa dạng khác. Tiêu biểu như:

1.2.1 Y tế

Một số công ty khởi nghiệp đã và đang sử dụng công nghệ VR để tạo ra các ca phẫu thuật "ảo". Nhờ đó các bác sĩ có thể nâng cao tay nghề mà không cần phải phẫu thuật trực tiếp trên người bệnh. Những ca phẫu thuật "ảo"này không những mô phỏng được thị giác, thính giác (hình ảnh, âm thanh khi phẫu thuật), mà còn mô phỏng được xúc giác (cảm giác khi cầm dao; tiếp xúc xương, da người bệnh,...) [6].

Ví dụ tiêu biểu cho công ty Fundamental VR vừa kêu gọi vốn thành công 20 triệu đô vào đầu tháng 8 năm 2022 [6].

Hình 1.3: Sản phẩm của công ty FundamentalVR để mô phỏng lại các ca phẫu thuật xương

1.2.2 Kiến trúc

VR được các công ty xây dựng ứng dụng nhằm dựng lên mô hình của căn nhà trước khi tiến hành xây dựng. Chủ nhân tương lai của căn nhà có thể trải nghiệm nội thất, kiến trúc của căn nhà một cách chân thực cũng như có thể đóng góp trực tiếp ý kiến để thay đổi thiết kế căn nhà trước khi tiến hành xây dựng [9].

1.2.3 Quân sự

Trong lĩnh vực quân sự VR được sử dụng để tạo ra các cuộc tập trận giả một cách chân thực nhất để huấn luyện cho binh sĩ. Điều này sẽ phần nào thay thế các cuộc tập trận bình thường gây tiêu hao rất nhiều đạn dược, vũ khí [2].

1.2.4 Du lịch

Công nghệ VR giúp tạo ra các video về cảnh quan ở các địa điểm du lịch. Khi đeo kính VR vào người dùng sẽ nhìn thấy không gian của địa điểm du lịch hiện ra ngay trước mắt. Không chỉ vậy, người dùng còn có thể di chuyển trong không gian này, nghe được các âm thành vòm 3D chân thực nhất [10].

1.3 Lý thuyết, cơ chế trong VR

1.3.1 Tính "đắm chìm"trong VR

Tính "đắm chìm"thể hiển độ chân thực của không gian mà thiết bị VR tạo ra cho người dùng. Tính "đắm chìm"càng cao, người dùng càng có cảm giác như được sống trong một thể giới khác. Nó được thể hiện qua các yếu tố dưới đây khi sử dụng thiết bị VR:

- Thị giác: không gian người dùng nhìn thấy.
- Thính giác: âm thanh vòm người dùng nghe được.
- Xúc giác: phản hồi khi tương tác với sự vật trong thế giới VR.

- Di chuyển và thăng bằng: cảm nhận kết hợp từ nhiều giác quan cùng với tiền đình để tạo ra cảm giác di chuyển.
- Khứu giác: mùi người dùng ngửi thấy.
- Vị giác: hương vị lưỡi cảm nhận được.

Hiện tại, công nghệ VR chỉ đang tập trung mang lại cảm giác chân thực nhất ở 4 yếu tố thị giác, thính giác, xúc giác, di chuyển và thăng bằng; trong đó thị giác là yếu tố trọng tâm nhất do đây là các yếu tố được con người sử dụng nhiều nhất để cảm nhận thế giới xung quanh.

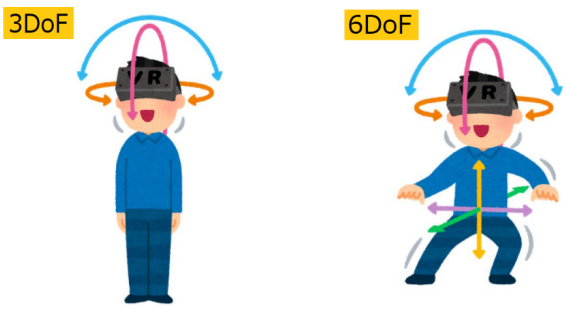
1.3.2 Bậc tự do

Khái niệm bậc tự do trong tiếng Anh là degree of freedom (DoF), nó hiển khả năng di chuyển của người dùng trong môi trường VR: mỗi bậc tự do tương ứng với một trục người dùng có thể di chuyển. Ví dụ, khi ta tiến lên một bước thì các vật ở trước mặt gần lại thêm 1 bước chứ chúng không bị dịch chuyển lùi về phía sau. Có tổng cộng 6 trục:

- Chuyển động quay: ứng với 3 trục khi người dùng
  - Quay đầu sang trái hay phải.
  - Ngẩng đầu lên hay xuống.
  - Nghiêng đầu sang trái hay phải.
- Chuyển động tịnh tiến: ứng với 3 trục khi người dùng
  - Di chuyển tiến lên, lùi xuống.
  - Di chuyển sang trái, sang phải.
  - Nhảy lên, cúi xuống.

Hiện tại các thiết bị kính VR thường được chia làm 2 loại:

- 3-DoF: Hỗ trợ 3 bậc tự do của chuyển động quay hoặc chuyển động tịnh tiến. Ví dụ: Google Cardboard.
- 6-DoF: Hỗ trợ cả 6 bậc tự do của chuyển động quay và chuyển động tịnh tiến. Ví dụ: Oculus Quest 2.



Hình 1.4: 3 DoF vs 6 DoF

1.4 Hạn chế của VR

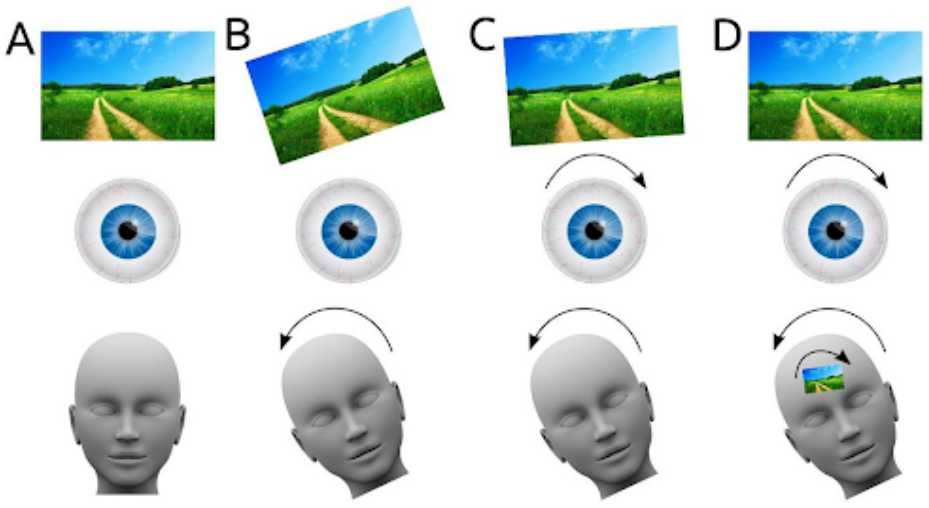
1.4.1 Hạn chế môi trường

Mặc dù chuyển động trong 6 bậc tự do có thể mô phỏng lại hết tất cả các hoạt động, VR không thể mô phỏng được không gian mà hoạt động đó diễn ra. Ví dụ như khi chạy bộ cần đường dài, khi leo núi cần vách đá cao dựng đứng,... phần lớn các người dùng VR sẽ ở trong một không gian chật hẹp hơn rất nhiều (như trong phòng, trong nhà) so với các không gian kể trên. Tất nhiên vẫn có cách để người dùng thực hiện các hoạt động này trong VR mà chỉ cần di chuyển tại chỗ nhưng chắc chắn nó sẽ làm giảm sự đắm chìm vì vị trí người dùng không thay đổi. Cho dù ta có biến đổi không gian ảo như thế nào đi nữa thì cũng không thể thay đổi không gian thực nơi mà người dùng thực sự di chuyển.

1.4.2 Say thực tế ảo

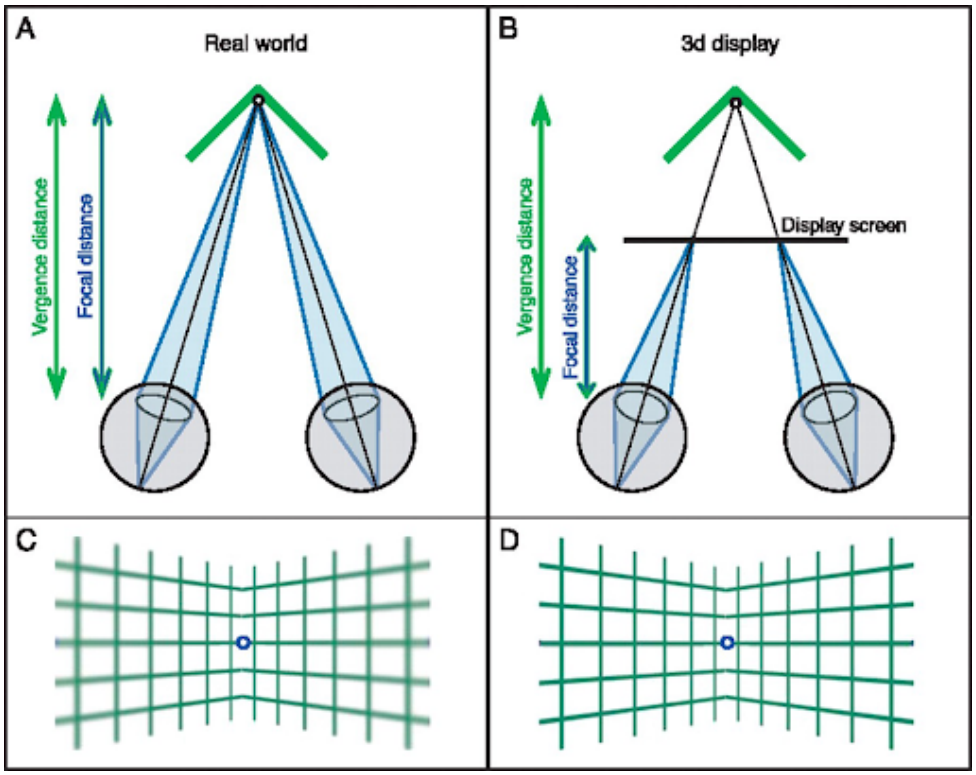
Chúng ta thường nghe thấy mọi người nói về say xe, say sóng hay say tàu, trong VR cũng có một khái niệm khá tương tự là say thực tế ảo. Hiện tượng này có thể xảy ra do nhiều nguyên nhân như tần số quét màn hình thấp làm cho sự di chuyển bị ngắt quãng hay sử dụng VR quá nhiều làm nhức mỏi mắt. Trong số này, có hai nguyên nhân lớn nhất bao gồm:

- **Hình ảnh mâu thuẫn với cảm nhận của tiền đình**: Tiền đình là cơ quan duy trì thăng bằng cho cơ thể. Khi chúng ta di chuyển, cúi, xoay người,... tiền đình sẽ rung lắc theo để đảm bảo cân bằng cho cơ thể. Khi mắt nhìn thấy không gian ảo xung quanh chuyển động nhưng tiền đình không cảm nhận được chuyển động này hoặc ngược lại, nó sẽ tạo ra mâu thuẫn, dẫn đến say thực tế ảo [5].



Hình 1.5: Hình ảnh mâu thuẫn với cảm nhận của tiền đình (Visual-vestibular mismatch)

- **Tiêu điểm mắt mâu thuẫn với vị trí vật thể**: Với các vật thể trong thế giới thực, ánh sáng từ chúng sẽ truyền thẳng đến mắt chúng ta. Nhưng trong thực tế ảo, ánh sáng sẽ đến từ màn hình mà người dùng đang quan sát, dẫn đến việc có mâu thuẫn giữa ảnh ở tiêu điểm mắt và vị trí của vật thể, như khi tiêu điểm mắt ở gần nhưng hình ảnh vật thể lại ở xa hoặc ngược lại



Hình 1.6: Tiêu điểm mắt mâu thuẫn với vị trí vật thể (Vergence-accommodation conflict)

1.4.3 Cách khắc phục

Với công nghệ hiện tại ta đang có một số cách khắc phục các hạn chế trên:

1. **Hạn chế môi trường** : sử dụng các thiết bị chuyên dụng để có thể mô phỏng lại các chuyển động mà không thực sự làm thay đổi vị trí người dùng. Phương pháp khá tốn kém, không thể áp dụng cho phần lớn người dùng VR.

2. **Say thực tế ảo** : sử dụng các chiến lược di chuyển khác nhau, tránh cách di chuyển truyền thống. Các chiến lược này bao gồm: teleport (dịch chuyển tức thời đến một vị trí), dùng các nút để di chuyển, snap turning (đổi góc nhìn bằng các giá trị rời rạc, không liên tục),...

2 Ý tưởng cho sản phẩm

Khởi đầu, nhóm chúng em muốn tìm một ý tưởng độc đáo mà chưa sản phẩm nào trên thị trường có thể đáp ứng. Việc này rõ ràng là khó vì, nếu ý tưởng đó quá dễ để thực hiện, thì khả năng cao là đã có một sản phẩm nào đó trên thị trường dựa trên ý tưởng đó rồi. Ngược lại, nếu ý tưởng quá khó, thì trong thời gian cho phép, chúng em sẽ không thể hoàn thành kịp tiến độ được.

Cho nên, chúng em bắt đầu hướng tới đối tượng đặc thù hơn: người Việt Nam. Nhóm không giới hạn ý tưởng trong lĩnh vực cụ thể nào, tuy nhiên, là một nhóm bài tập lớn làm về VR, bọn em muốn sản phẩm có thể tận dụng được những lợi thế của VR, ngoài ra sản phẩn còn có thể tích hợp được những công nghệ hiện đại trong công nghệ thông tin. Và cuối cùng, bọn em tìm được một chủ đề có thể áp dụng những tiêu chí đó, đó là

phát triển những công nghệ hiện đại cho một trò chơi dân gian của Việt Nam có tên là *cờ tướng*.

Cờ tướng, qua tìm hiểu của bọn em, hiện không có sản phẩm nào áp dụng công nghệ VR mà có giao diện Tiếng Việt. Thậm chí, cũng không có sản phẩm cờ tướng VR nào hỗ trợ giao diện Tiếng Anh, mà chỉ có 1 game cờ tướng VR duy nhất hỗ trợ tiếng Trung [3]. Lý do bởi vì, game cờ tướng chỉ phổ biến ở Trung Quốc và Việt Nam chứ không phổ biến rộng rãi như game cờ vua. Chúng em càng có thêm động lực để xây dựng một game cờ tướng VR đầu tiên dành cho người Việt, nhằm cải tiến trò chơi dân gian này.

Ngoài ra, bọn em cũng muốn tăng cường trải nghiệm VR cho sản phẩm này. Vì khi chơi cờ tướng trong môi trường thực tế ảo, độ chính xác cho thao tác di chuyển quân cờ bằng tay không cao, nên chúng em muốn xây dựng một cơ chế "rảnh tay". Đó là lúc bọn em nghĩ ra ý tưởng về game cờ tướng VR cho phép điều khiển quân cờ bằng giọng nói.

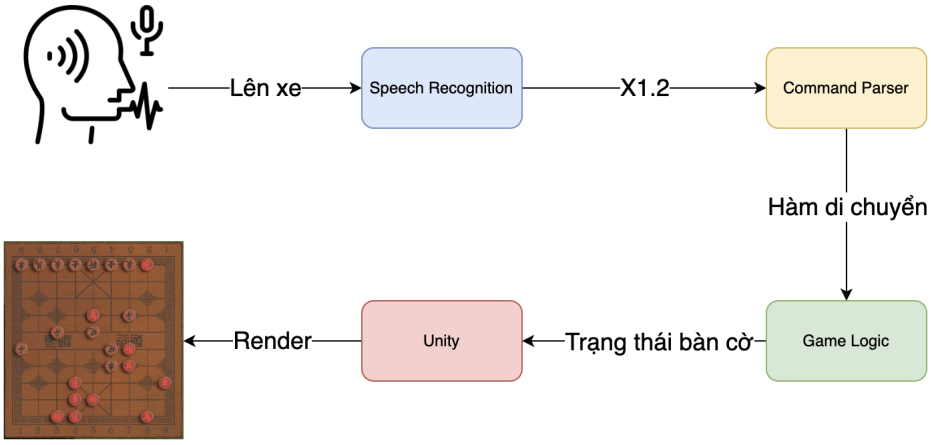
3 Cài đặt sản phẩm

Trước khi cài đặt sản phẩm, nhóm chúng em và nhóm AI cho cờ tướng đã cùng nhau xây dựng một chuẩn cờ tướng với các tiêu chí:

- Tương thích với chuẩn cờ tướng của Liên đoàn cờ tướng Việt Nam
- Tập lệnh gần với logic tự nhiên, dễ dàng để điều khiển bằng giọng nói
- Có thể mở rộng
- Dễ dàng cài đặt

Và bọn em đã hoàn thành việc xây dựng một chuẩn cờ tướng chung, có thể dễ dàng cài đặt cho game cờ tướng này, và cũng có thể được áp dụng cho các game cờ tướng khác. **Chuẩn được đăng tải tại địa chỉ *uet-cacvdhd.github.io/xiangqi-docs***. Việc xây dựng được chuẩn cờ tướng là bước đầu tiên và cũng là nền tảng để bọn em có thể cài đặt được sản phẩm cờ tướng này.

Hình dưới đây mô tả kiến trúc game với 4 thành phần chính sẽ được trình bày chi tiết ở các phần sau:



Hình 3.1: Các thành phần chính trong game

3.1 Game Logic - Logic chính của game

Trong phần này chúng em xin trình bày về logic chính của game bao gồm các tác vụ liên quan đến quản lý trạng thái ván cờ (đang chơi, chiếu tướng, hết cờ); kiểm tra, hiển thị các nước có thể đi được của quân cờ,... Điểm đặc biệt của thành phần này trong game là nó không bị phụ thuộc vào các class có trong Unity, có thể tách thành phần này và để áp dụng cho game cờ tướng được phát triển trên các nền tảng khác, engine game khác.

3.1.1 Biểu diễn toạ độ bàn cờ

Yêu cầu

- Biểu diễn vị trí của các quân cờ trên bàn cờ.
- Tính toán toạ độ x, y, z trong Unity của các quân cờ.

Giải pháp

Nhóm định nghĩa ra 2 hệ toạ độ:

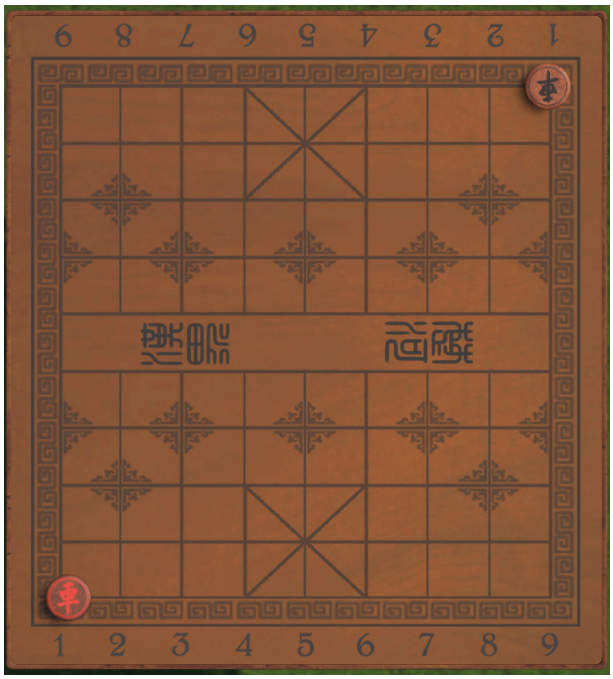
- Hệ toạ độ Oxyz: hệ toạ độ Oxyz của các vật thể trong Unity.
- Hệ toạ độ bàn cờ: vị trí của quân cờ trên bàn cờ. 9 cột trên bàn cờ được đánh số từ 1 đến 9 tính từ trái sang phải, các hàng trên bàn cờ được đánh số từ 1 đến 10 tính từ dưới lên trên.

Việc chuyển đổi từ toạ độ bàn cờ sang toạ độ Oxyz trong Unity sẽ được trình bày chi tiết ở phần 3.4.1.

Khi xử lý các logic liên quan đến gì chuyển của quân cờ, ví dụ qua sông, với bên đỏ một quân cờ được tính là qua sông khi nó ở hàng $>= 6$ còn với bên đen là ở hàng $<= 5$. Như vậy khi xử lý các logic tương tự, sẽ bị lặp code và phải dùng nhiều lệnh *if, else*. Do đó trong hệ toạ độ bàn cờ, nhóm lại tiếp tục chia thành 2 loại:

- Hệ toạ độ bàn cờ tương đối: có 3 trục (hàng, cột, màu quân cờ). Hệ toạ độ này dùng để xử lý vấn đề nói trên. Khi xử lý các logic như qua sông, quân cờ của cả 2 bên đều được tính là qua sông khi chỉ số hàng $>= 6$.
- Hệ toạ độ bàn cờ tuyệt đối: có 2 trục (hàng, cột). Hệ toạ độ này dùng để lưu trạng thái bàn cờ trong mảng 2 chiều (chi tiết hơn ở phần 3.1.2).

Lấy ví dụ 2 quân xe của 2 bên đỏ và đên nằm ở 2 góc bàn cờ như hình dưới đây:



Hình 3.2: Các cột trên bàn cờ được đánh số từ 1 -> 9

- Trong hệ toạ độ bàn cờ tương đối, 2 quân cờ này có toạ độ là (1, 1, đỏ) và (1, 1, đen).
- Trong hệ toạ độ bàn cờ tuyệt đối, 2 quân cờ này có toạ độ là (1, 1) và (10, 9).

3.1.2 Load, restart trạng thái bàn cờ

Yêu cầu

Game cần hỗ trợ các tính năng sau.

- Load - Lưu: sao lưu lại trạng thái bàn cờ để chơi tiếp vào thời điểm khác.
- Restart - Chơi mới: chơi từ đầu một ván cờ.

Giải pháp

Biểu diễn trạng thái bàn cờ tại thời điểm chơi bằng một class có tên *GameSnapShot*. Class này có một số trường sau:

- sideTurn: thông tin về lượt chơi hiện tại là của bên đỏ hay đen.
- state: thông tin về trạng thái bàn cờ (chi tiết ở phần 3.1.5).
- chessboard: mảng 2 chiều với kích thước 10 x 9 tượng trưng cho các ô trên bàn cờ. Nếu ở một ô trên bàn cờ chứa quân cờ thì phần tử tương ứng trong mảng sẽ chứa con trỏ đến đối tượng quân cờ.

Khi người dùng chọn lưu ván cờ, đối tượng thuộc class *GameSnapShot* sẽ được serialize về định dạng json và ghi ra file. Khi người chơi muốn chơi tiếp ván chơi thì file này sẽ được đọc và một đối tượng mới thuộc class *GameSnapShot* sẽ được khởi tạo.

```
{
  "sideTurn": "Red",
  "state": "Playing",
  "chessPieces": [
    {
      "aCell": {
        "col": 3,
        "row": 1
      },
      "isDead": false,
      "side": "Red",
      "type": "T"
    },
    {
      "aCell": {
        "col": 4,
        "row": 1
      },
      "isDead": false,
      "side": "Red",
      "type": "S"
    },
    ...
  ]
}
```


Hình 3.3: Cấu trúc file json lưu trạng thái game

Ngoài ra, nhóm tạo một file lưu lại trạng thái bàn cờ khi bắt đầu một ván cờ để khi người chơi chọn chơi mới thì sẽ khởi tạo đối tượng thuộc class *GameSnapShot* từ file này thay vì đọc file trạng thái ván cũ như trên.

3.1.3 Hiển thị khả năng di chuyển của quân cờ

Yêu cầu

Khi người chơi click chọn một quân cờ (chế độ chơi thường của game) hay khi nhấc quân cờ lên (chế độ chơi VR của game) các vị trí quân cờ này có thể di chuyển tới sẽ được hiển thị.

Giải pháp

Chúng em định nghĩa một số khái niệm sau:

- *Hướng đi*: các hướng quân cờ có thể đi. Có tất cả 8 hướng, khi đang ở ô (i, j) và di chuyển theo hướng này, vị trí mới của quân cờ là:
  1. Up: (i + 1, j)
  2. Right: (i, j + 1)
  3. Down: (i - 1, j)
  4. Left: (i, j - 1)

5. UpRight: (i + 1, j + 1)
6. DownRight: (i - 1, j + 1)
7. DownLeft: (i - 1, j - 1)
8. UpLeft: (i + 1, j - 1)

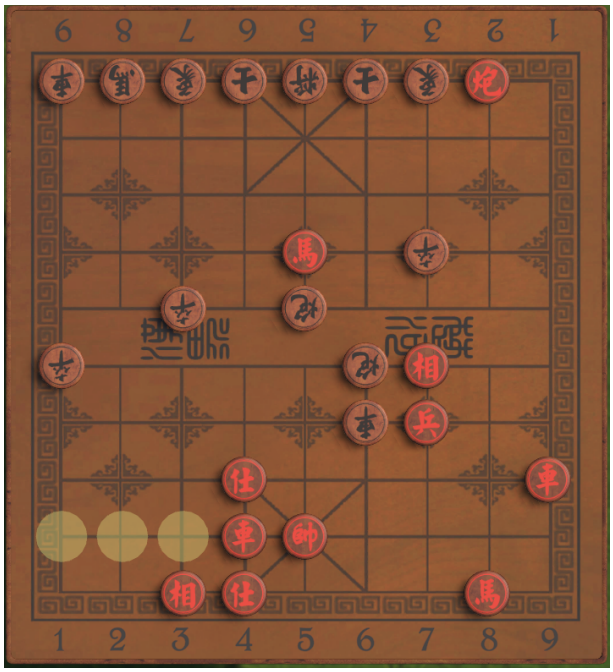
- *Đường đi*: là cách kết hợp các hướng đi.
- *Nước đi*: là việc lặp lại một số lần **một** đường đi.

Ngoài ra, khi quân cờ thực hiện nước đi, cần kiểm tra khi thực hiện nước đi này quân cờ có gặp vật cản (quân cờ khác) trên đường từ vị trí hiện tại tới vị trí đích hay không. Nếu có đúng một vật cản và vật cản nằm ở vị trí đích, vật cản là quân cờ của đối phương thì có thể "ăn", nước đi tính là hợp lệ. Ngược lại, nước đi này tính là không hợp lệ.

Ví dụ

Với quân xe:

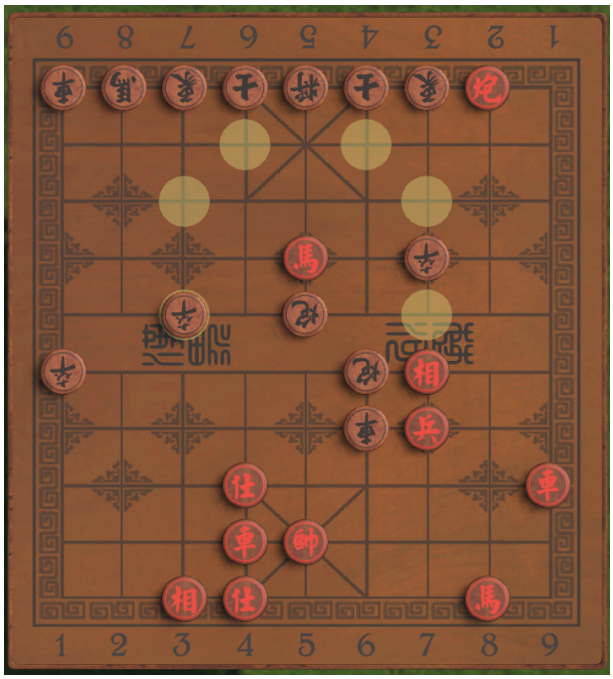
- Các đường đi hợp lệ:
  1. Up
  2. Right
  3. Down
  4. Left
- Các nước đi hợp lệ (nếu không có vật cản):
  - Lặp lại đường đi Up, Down tối đa 10 lần (số hàng trên bàn cờ).
  - Lặp lại đường đi Left, Right tối đa 9 lần (số cột trên bàn cờ).



Hình 3.4: Gợi ý nước đi của quân xe

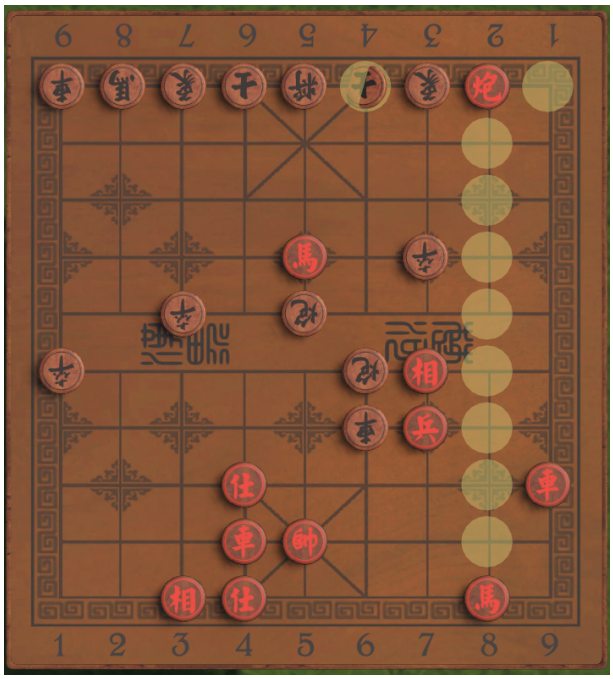
Với quân mã:

- Các đường đi hợp lệ:
  1. Up, UpLeft
  2. Up, UpRight
  3. Right, UpRight
  4. Right, DownRight
  5. Down, DownLeft
  6. Down, DownRight
  7. Left, DownLeft
  8. Left, UpLeft
- Các nước đi hợp lệ (nếu không có vật cản):
  - Lặp lại tất cả các đường đi tối đa một lần.



Hình 3.5: Gợi ý nước đi của quân mã

Trường hợp đặc biệt: đối với quân pháo - quân cờ có khả năng "nhảy"qua đầu một quân cờ khác để ăn quân cờ của đối phương, cần kiểm tra nếu trên đường đi của quân pháo có 2 vật cản, trong đó 1 vật cản nằm ở vị trí đích và là quân cờ của đối phương thì nước đi được coi là hợp lệ.



Hình 3.6: Gợi ý nước đi của quân pháo

3.1.4 Không hiển thị các nước đi có thể dẫn đến thua cuộc

Yêu cầu

Không hiển thị gợi ý các nước đi có thể dẫn đến 1 trong 2 trường hợp bên di chuyển bị thua sau:

1. Khi di chuyển xong, quân tướng của bên di chuyển có thể bị ăn bởi đối phương.
2. Khi di chuyển xong, quân tướng sẽ "đối mặt"với quân tướng đối phương (2 quân tướng nằm cũng cột và giữa 2 quân tướng không có quân cờ nào).

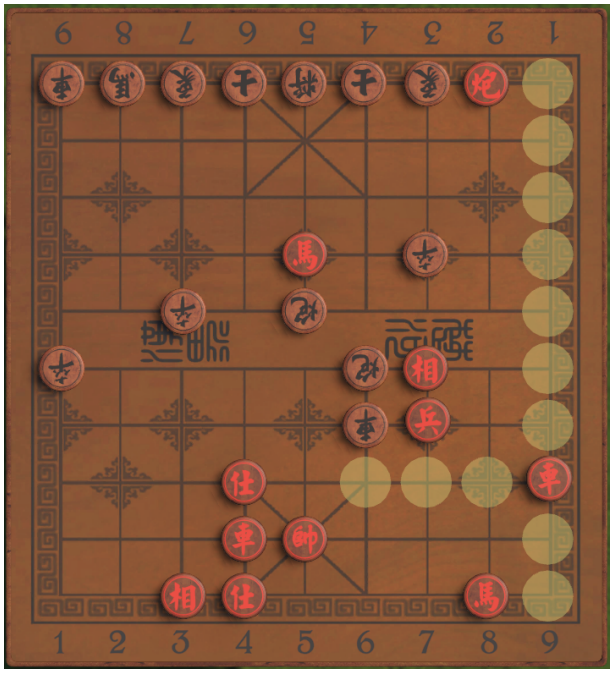
Giải pháp

Sau khi tìm được danh sách các ô hợp lệ quân cờ có thể di chuyển đến như ở phần 3.1.3, với từng ô (i, j) trong danh sách này:

- Tạo một đối tượng thuộc class *GameSnapShot* mới, giống hệt đối tượng biểu diễn trạng thái bàn cờ hiện tại trừ việc quân cờ đang chọn sẽ có vị trí mới là (i, j).
- Kiểm tra xem quân tướng của bên vừa thực hiện di chuyển có thể bị ăn hoặc "đối mặt"với quân tướng của đối phương hay không. Nếu có, thì không hiển thị gợi ý nước đi này cho người chơi.

Ví dụ

Trong hình dưới đây, game không hiển thị gợi ý cho quân xe đi vào ô (3, 5) bởi khi di chuyển vào ô này con pháo ở ô (6, 5) của bên đen có thể "ăn"tướng bên đỏ.



Hình 3.7: Gợi ý nước đi không dẫn đến thua cuộc của quân xe

3.1.5 Kiểm tra trạng thái ván cờ

Yêu cầu

Game cần kiểm tra và hiển thị thông báo trên giao diện khi:

- Người chơi bị chơi bị chiếu tướng.

- Người chơi bị chiếu hết cờ và thua.

Giải pháp

Nhóm định nghĩa các 3 trạng thái có thể xảy ra của một ván cờ:

1. *Playing*: Trạng thái bình thường.
2. *Checkmate*: Trạng thái khi bên đến lượt di chuyển bị chiếu tướng.
3. *GameOver*: Trạng thái khi bên đến lượt di chuyển bị chiếu hết cờ.

Ngay khi một người chơi, giả sử là bên Đen thực hiện di chuyển quân cờ và đến lượt bên Đỏ, một hàm sẽ được thực thi kiểm tra con tướng của bên Đỏ có thể bị ăn không (tồn tại một quân cờ bên Đen di chuyển được tới vị trí của tướng bên Đỏ)

- Nếu tướng Đỏ không thể bị ăn, trạng thái game vẫn là *Playing*.
- Ngược lại, trạng thái game chuyển sang *Checkmate*. Ngay sau đó, một hàm khác lại được thực thi đê kiểm tra xem bên Đỏ đã hết cờ chưa bằng cách xét khả năng di chuyển của tất cả quân cờ bên Đỏ. Nếu không một quân cờ nào bên Đỏ di chuyển được sang vị trí khác, bên Đỏ thua, trạng thái game chuyển sang *GameOver*.

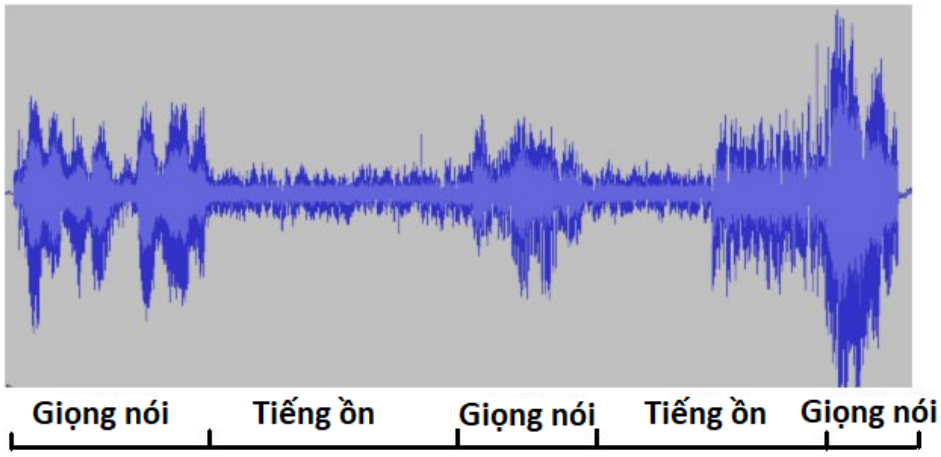
3.2 Speech Recognition - Nhận diện giọng nói

Trong phần này, chúng em sẽ trình bày về 2 bài toán chính nhóm em cần xử lý khi nhận diện giọng nói, đó là phát hiện hành động nói (voice activity detection) và chuyển từ giọng nói sang văn bản (speech-to-text).

3.2.1 Phát hiện hành động nói

Phát hiện hành động nói, hiểu đơn giản, là việc đánh dấu điểm đầu và điểm kết thúc của một câu lệnh nói. Trong game cờ tướng mà bọn em xây dựng, việc điều khiển quân cờ bằng giọng nói sẽ được thực thi qua các lệnh giọng nói. Tuy nhiên, đầu vào giọng nói mà game nhận được là một luồng liên tục. Để có thể đánh dấu điểm bắt đầu và điểm kết thúc, có hai phương pháp chính:

- Trực tiếp: Người dùng trực tiếp đánh dấu điểm bắt đầu và kết thúc câu lệnh, bằng một cơ chế, có thể kể đến là cơ chế bộ đàm — push-to-talk. Trong cơ chế này, người dùng nhấn và giữ một nút và bắt đầu nói để game bắt đầu thu âm. Kết thúc câu lệnh, người dùng nhả nút đó ra để game ngừng thu âm. Câu lệnh được bắt đầu từ khi game bắt đầu thu âm và kết thúc khi game ngừng thu âm. Dễ thấy rằng cơ chế này sẽ gây bất tiện cho người chơi, vì mỗi lệnh giọng nói sẽ liên tục phải bấm nút để bắt đầu và kết thúc.
- Gián tiếp: Game sẽ có cơ chế phát hiện cường độ âm lượng, tần số,... để tự động phát hiện thời điểm người chơi bắt đầu nói, và thời điểm người chơi dừng nói. Phát hiện hành động nói theo cách gián tiếp này là một trong những bài toán đã được cụ thể hoá trong lĩnh vực xử lý giọng nói, với tên gọi là *voice activity detection* (VAD). Đây là phương pháp phát hiện giọng nói mà game chúng em sử dụng, và sẽ được trình bày cụ thể trong phần này.



Hình 3.8: Một lệnh giọng nói được biểu diễn dưới dạng độ lớn của âm

Chúng em tham khảo qua các công nghệ hiện đại nhất trong lĩnh vực này và tìm ra được các giải pháp như "A robust frontend for VAD: exploiting contextual, discriminative and spectral cues of human voice"có sử dụng kết hợp nhiều mô hình deep learning, hay "VAD trong WebRTC", là cơ chế phát hiện giọng nói phổ biến nhất bây giờ, được sử dụng rộng rãi trong các ứng dụng phòng họp trực tuyến hiện nay. Các thuật toán nêu trên đều cho ra khả năng phát hiện tương đối chính xác (trên 70% [1]), tuy nhiên nhược điểm là thuật toán quá phức tạp nên khó để cài đặt cho dự án game này. Một giải pháp khác là sử dụng thư viện C# cho xử lý VAD, bọn em cũng đã cân nhắc để sử dụng, nhưng quyết định cuối cùng là, bọn em thử implement một cơ chế đơn giản để xử lý giọng nói, nếu nó không đủ tốt, thì bọn em sẽ dùng các thư viện có sẵn này.

Cụ thể về cơ chế này, đầu tiên bọn em sẽ định nghĩa các thành phần đầu vào và đầu ra:

- Đầu vào: dữ liệu giọng nói được lấy mẫu tần suất 16000 Hz, 1 kênh, được biểu diễn dưới dạng một mảng biểu diễn lần lượt các mẫu được lấy trong một khoảng thời gian, với các giá trị là số thực trong khoảng [-1, 1]. Ví dụ, mảng được lấy mẫu có thể có giá trị là ['0.001', '0.008', '0.524', '0.048', '0.842', '0.470', '0.056', '0.731', '0.002', '0.001'].
- Đầu ra: chỉ số trong mảng khi hành động nói bắt đầu và khi hành động nói kết thúc.

Với đầu vào là mảng như trên, chúng em xây dựng thuật toán phát hiện người chơi có đang nói hay không với cơ chế đơn giản nhất là phát hiện độ to của âm thanh. Thuật toán sử dụng một cửa sổ lấy mẫu âm thanh $sampleWindow$ với độ dài n. Liên tục mỗi lần mẫu âm thanh được cập nhật, cửa số lấy mẫu $sampleWindow$ sẽ di chuyển đến vị trí mới nhất và thuật toán sẽ so sánh tổng trung bình các mẫu trong cửa số với một ngưỡng $thresh$ được đặt ra, nếu lớn hơn $thresh$ thì tức là đang có âm thanh giọng nói ở thời điểm hiện tại, ngược lại thì không. Ngoài ra, thuật toán sử dụng một đồng hồ $timer$ để giữ cho trạng thái nói không được chuyển sang trạng thái ngừng nói trong thời gian t, để đảm bảo rằng, khi người nói ngắt nghỉ giữa các từ trong lệnh, thuật toán không coi khoảng

ngắt nghỉ đó là kết thúc của lệnh.

$$havingVoice = \frac{\sum_{i=0}^{n-1} sampleWindow[i]}{n} \geq thresh$$

Nếu $havingVoice$ = true, reset $timer$ về giá trị 0. Người dùng là đang nói ($isSpeaking$) nếu giá trị hiện tại của $timer < t$, ngược lại, người dùng đang không nói. Frame bắt đầu nói được tính từ khi $isSpeaking$ có giá trị $true$ và frame kết thúc là khi $isSpeaking$ chuyển sang giá trị false. Vì khi bắt đầu nói, thuật toán sẽ không phát hiện khi người dùng cất lên giọng mà nó phát hiện khi giọng người dùng vừa đủ để vượt qua ngưỡng $thresh$, cho nên frame bắt đầu nói sẽ cần phải trừ đi một số lượng frame là $rollback$ frame.

Với thuật toán phát hiện giọng nói này, bọn em thử trong nhiều môi trường có độ ồn nhất định và nhận thấy thuật toán chạy cho ra kết quả tương đối chính xác.

3.2.2 Chuyển từ giọng nói sang văn bản

Vấn đề tiếp theo sau khi trích xuất được đoạn âm thanh có chứa lệnh, là xử lý như thế nào để biến đoạn âm thanh đó thành văn bản. Bọn em không xử lý phần âm thanh này, mà gửi đoạn âm thanh này đến server do nhóm xử lý giọng nói game cờ tướng (nhóm 2) xây dựng. Chi tiết cơ chế xử lý giọng nói có thể tham khảo tài liệu của nhóm xử lý giọng nói game cờ tướng.

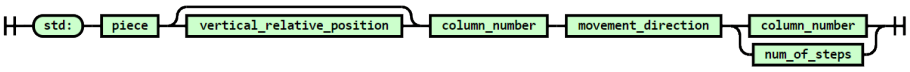
Cuối cùng, server xử lý giọng nói sẽ trả về text là các token parser được từ giọng nói để nhóm chúng em parse ra lệnh chạy được trong Unity.

3.3 Command Parser - Xử lý cú pháp các lệnh

Sử dụng chuẩn cờ tướng nhóm em và nhóm xử lý giọng nói cho game cờ tướng đã xây dựng nên, để game có thể hiểu được các lệnh đó, thì cần phải có một cơ chế parse (phân tích cú pháp) lệnh sang các đối tượng Unity. Để có thể parse các lệnh một cách đơn giản thì chúng ta nên xây dựng một cây cú pháp trừu tượng (abstract syntax tree - AST). Cây này cho phép đi đến mọi khả năng của lệnh một cách đơn giản, và cho phép quay lui nếu lựa chọn dẫn đến lệnh không thể parser. Ngoài ra, sử dụng kiểu dữ liệu cây, các thành phần của cây có thể được tái sử dụng và lắp ghép lại vào nhau một cách tiện lợi.

Việc xây dựng cây cú pháp trừu tượng sao cho dễ dàng tái sử dụng và đảm bảo logic chặt chẽ, cũng như hiệu năng cao là một bài toán khó. Chính vì vậy, đã có rất nhiều thư viện được sinh ra để xây dựng các cây cú pháp này, và giúp dễ dàng tái sử dụng lại nhiều thành phần của cây, và còn có thể kết hợp với các logic đại số Boole. Và nhóm chúng em đã sử dụng thư viện Parlot, để biểu diễn chuẩn cờ tướng theo chuẩn PEG (Parsing expression grammar) mà chúng em xây dựng bằng các dòng lệnh.

Sử dụng thư viện này, chúng em chỉ cần định nghĩa các thành phần cơ bản nhất, sau đó ghép nối lại bằng các thành phần Or, And, ZeroOrOne, OneOf,... thư viện sẽ tự xây dựng nên một cây AST, và tạo thành một đối tượng parser. Sau đó chúng ta có thể sử dụng đối tượng parser để parse mọi loại xâu chúng ta muốn, và sẽ nhận được một object trích xuất từ các thành phần trong xâu đó, hoặc là nhận được lỗi nếu xâu không hợp lệ.



Hình 3.9: Parser cho lệnh chuẩn theo luật cờ tướng Việt Nam

Parser của chúng em được định nghĩa gồm các thành phần parser con là:

- Parser lệnh di chuyển chuẩn: Trong lệnh di chuyển, các cú pháp của lệnh giống với cú pháp của chuẩn cờ tướng Việt Nam. Parser có thể hiểu các lệnh phổ thông, được chuẩn hoá trong luật cờ tướng Việt Nam như: "mã 3 bình 5", "tốt trước giữa 2 tiến 1", "pháo 3 trước tiến 2", "xe 5 bình 4"
- Parser lệnh di chuyển mở rộng: Đây là một hệ thống lệnh do bọn em xây dựng, kế thừa từ tập lệnh di chuyển chuẩn. Lệnh này mở rộng cách chỉ định quân cờ bằng các hướng trái phải, cho phép di chuyển theo hướng chéo hay 1 hướng nhất định sang ngang nhất định (sang trái, sang phải, tiến trái, lùi trái,...). Lệnh cũng cho phép không cần nói đầy đủ các tham số như lệnh di chuyển chuẩn. Khi người dùng không nói đầy đủ tham số, logic game sẽ tự động tìm các quân cờ và các nước đi thoả mãn điều kiện nói đó. Nếu số lượng các nước đi thoả mãn = 1, lệnh sẽ tự động được thực thi. Điều này giúp các lệnh này có thể được thực hiện: tướng lên, tướng sang trái, pháo trái tiến,...
- Parser lệnh ăn quân: Đây cũng là một hệ thống lệnh mở rộng do bọn em xây dựng, trong đó việc chỉ định nước đi đến được thay bằng chỉ định quân cờ muốn ăn. Ví dụ, có thể chỉ định: "tốt ăn mã", "pháo trái ăn xe trái", "ăn pháo",... Lệnh này giúp cho các lệnh giọng nói gần với ngôn ngữ tự nhiên hơn, gần với logic suy luận của con người hơn
- Parser lệnh chiếu tướng: Cũng nằm trong hệ thống lệnh mở rộng, lệnh chiếu tướng cho phép các lệnh như "chiếu tướng", "pháo trái chiếu tướng", "pháo phải chiếu tướng trực tiếp",...
- Parser lệnh qua sông: Lệnh qua sông cho phép chỉ định việc con tốt qua sông, ví dụ như "tốt trái qua sông", "tốt 3 qua sông"
- Parser lệnh meta: Đây là các lệnh không để điều khiển quân cờ trong game, nhưng đóng vai trò quan trọng trong việc điều khiển tổng thể cả bàn cờ. Những lệnh này bao gồm: "đầu hàng", "đi lại", "xin hoà", "chấp nhận", "từ chối", "chơi lại"...

3.4 Unity

Trong phần này, chúng em sẽ nói về việc sử dụng Unity trong dự án. Unity là một trong các game engine miễn phí phổ biến nhất có hỗ trợ việc phát triển các ứng dụng VR, với lựa chọn đáng cân nhắc khác là Unreal Engine. Tuy nhiên, Unreal Engine là công cụ để tạo ra game theo hướng tối ưu hiệu năng và đồ hoạ, và sử dụng ngôn ngữ lập trình C++ cho phép lập trình viên tương tác gần hơn với ngôn ngữ máy, sẽ tốn nhiều thời gian để phát triển và tối ưu hơn. Ngoài ra, cộng đồng làm game VR của Unreal Engine cũng nhỏ hơn, với ít tài nguyên có sẵn được chia sẻ hơn. Còn Unity có một lượng khóa học trực tuyến miễn phí nhiều gấp đôi so với Unreal Engine, bao gồm cả các khóa học về VR. Chính vì vậy, nhóm chúng em quyết định chọn Unity để phát triển ứng dụng này để đảm

bảo làm xong đúng tiến độ.

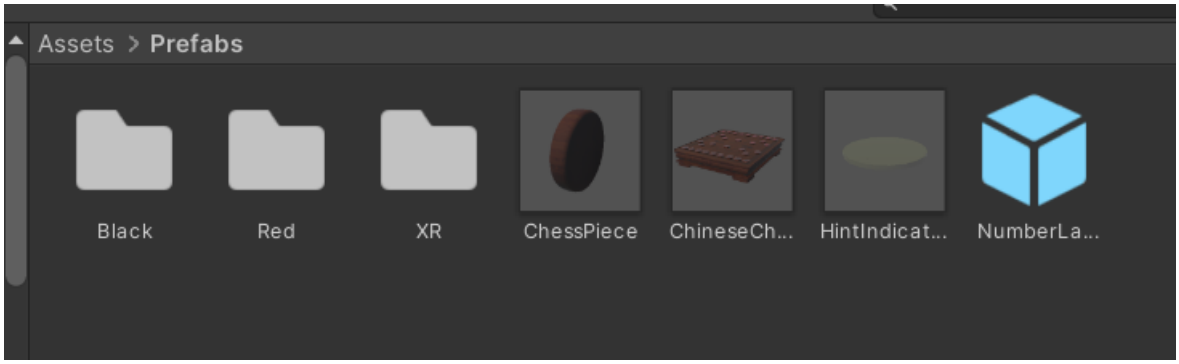
3.4.1 Logic model 3D

Đầu tiên ta cần phải làm rõ một thứ: Unity là công cụ **sử dụng** các model 3D, không phải để **tạo** ra chúng, mặc dù nó có hỗ trợ việc tạo ra các model nhưng đó không phải là thế mạnh của Unity. Do đó, nhóm chúng em sử dụng một công cụ khác là Blender chuyên về việc tạo này hơn để tạo ra các model quân cờ và bàn cờ rồi đưa vào trong Unity.

Bản thân model các quân cờ rất đơn giản, nó chỉ là một khối trụ, ngay cả bàn cờ cũng chỉ là một mặt phẳng hoặc khối hộp chữ nhật. Cái khó ở đây là texture của chúng, bao gồm màu sơn nền và chữ trên các quân cờ cũng như ô lưới trên bàn cờ. Việc vẽ các texture này rất tốn thời gian và làm cho dự án bị trì trệ, vậy nên để đẩy nhanh tiến độ nhóm chúng em đã sử dụng các texture có sẵn trên kho tài nguyên của Unity.

Prefab

Sau khi có các model với texture đầy đủ, ta sẽ biến đổi nó thành prefab. Prefab trong Unity là một tính năng đặc biệt cho phép lưu một model đã được cấu hình thành một tài nguyên có thể tái sử dụng được [8]. Khi một model đã được biến đổi thành prefab, ta chỉ việc kéo thả nó vào scene để dùng với số lượng không giới hạn mà không cần phải cấu hình lại. Chức năng này cực kỳ có ích khi ta sử dụng các model có định dạng FBX phải gán lại texture sau mỗi lần nhập mới. Nhưng điểm mạnh nhất của prefab nằm ở khả năng liên kết giữa các đối tượng thuộc cùng prefab, về cơ bản nếu một đối tượng được thay đổi, nó sẽ được cập nhật ở tất cả các đối tượng khác thuộc cùng prefab. Trong trò chơi cờ tướng, ngoại trừ con tướng ra thì tất cả các quân cờ còn lại đều có số lượng từ hai con trở lên, vậy nên khả năng liên kết này cực kỳ tiện mỗi khi cần cập nhật logic các quân cờ. Ta có thể tạo prefab rất dễ dàng bằng việc kéo thả model vào trong tệp prefab.



Hình 3.10: Tệp prefab

Trong dự án này, nhóm chúng em sử dụng 4 loại prefab:

- Quân cờ (bao gồm quân đỏ và đen): mỗi loại quân (xe, mã, pháo, ...) lại sử dụng một prefab riêng kế thừa prefab "quân cờ".
- Bàn cờ.
- Vòng tròn gợi ý dùng để biểu diễn các nước đi gợi ý.
- Nhãn gán số cột trên bàn cờ.

Tham chiếu tọa độ trong game ra tọa độ model

Một vấn đề quan trọng cần giải quyết trong trò chơi này là các quân cờ sẽ nằm ở đâu và di chuyển như thế nào trên bàn cờ, vì mặc dù ta có thể quy định tọa độ quân cờ trong logic game, Unity sẽ không thể sử dụng các tọa độ đó để cập nhật vị trí model. Vì vậy, chúng em đề ra một giải pháp là sử dụng ba Empty (một vật thể tương tự quân cờ nhưng là vật thể rỗng) tượng trưng cho ba điểm trên một mặt phẳng, trong đó một Empty sẽ có vai trò là điểm gốc. Từ ba điểm này ta có thể định nghĩa được hai vector là hai trục tọa độ cũng như độ dài cơ bản trên hai trục đó, khi đã có hai tham số là vector trục tọa độ và độ dài cơ bản, kết hợp với điểm gốc nữa là ta có thể suy luận ra mọi điểm trên mặt phẳng.

Gọi ba Empty này là $H_g(x_0, y_0, z_0)$, $H_x(x_1, y_1, z_1)$, $H_y(x_2, y_2, z_2)$ với H_g là điểm gốc thì ta sẽ có được:

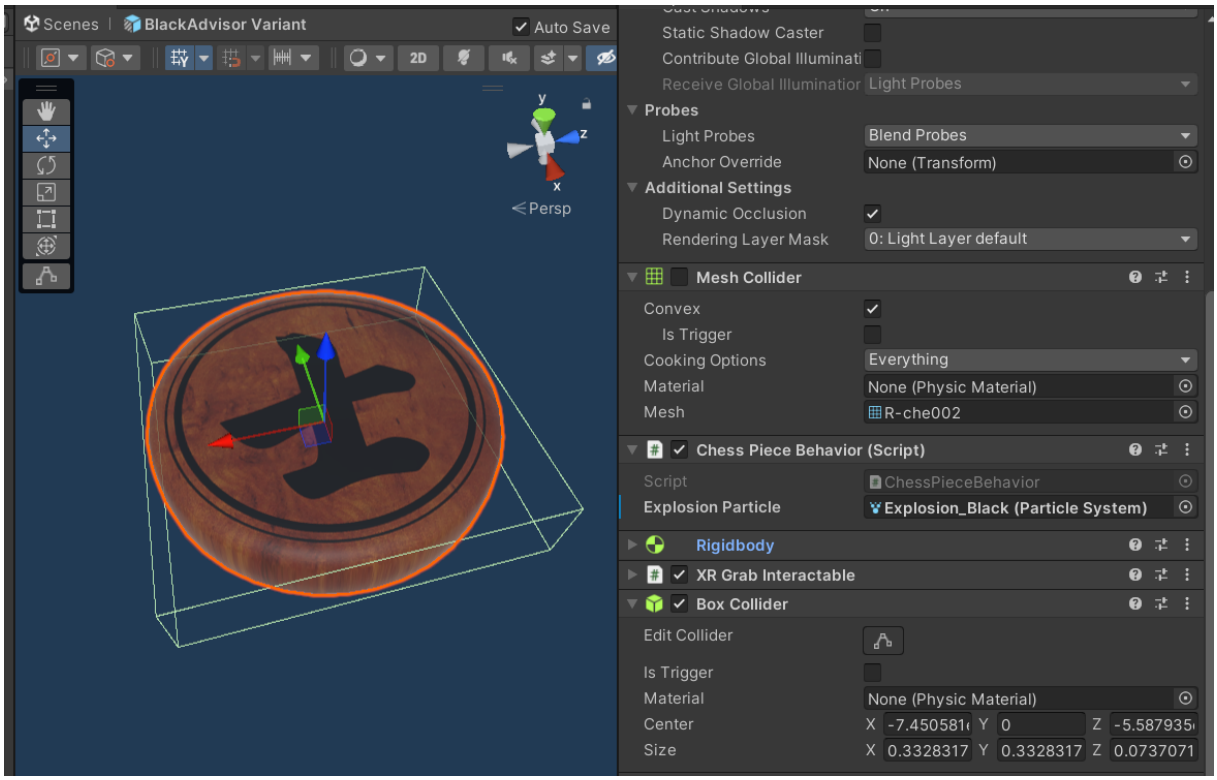
1. Vector trục tọa độ
    - Vector hướng x: $V_x = norm(H_g - H_x)$
    - Vector hướng y: $V_y = norm(H_g - H_y)$
2. Độ dài cơ bản
    - Độ dài cơ bản trục x: $d_x = |H_g - H_x|$
    - Độ dài cơ bản trục y: $d_y = |H_g - H_y|$

Từ đó ta có công thức tham chiếu từ tọa độ trong game (N_x, N_y) ra tọa độ model H_{new}:

$$H_{new} = H_g + V_x * d_x * N_x + V_y * d_y * N_y$$

Xử lý va chạm

Unity xử lý va chạm giữa các vật thể bằng các collider [8], về cơ bản chúng như một cái lưới bọc xung quanh vật thể nhưng không nhất thiết phải có hình dạng giống nó. Collider cũng có các hình dạng cơ bản giống model như hình hộp chữ nhật, hình cầu, hình viên thuốc,... Unity có hỗ trợ cả collider dạng lưới là collider có hình dạng giống với vật thể, được tạo ra bằng cách sinh tự động một lưới tam giác phỏng theo model của vật thể. Va chạm giữa hai vật thể xảy ra khi collider của chúng chạm vào nhau.

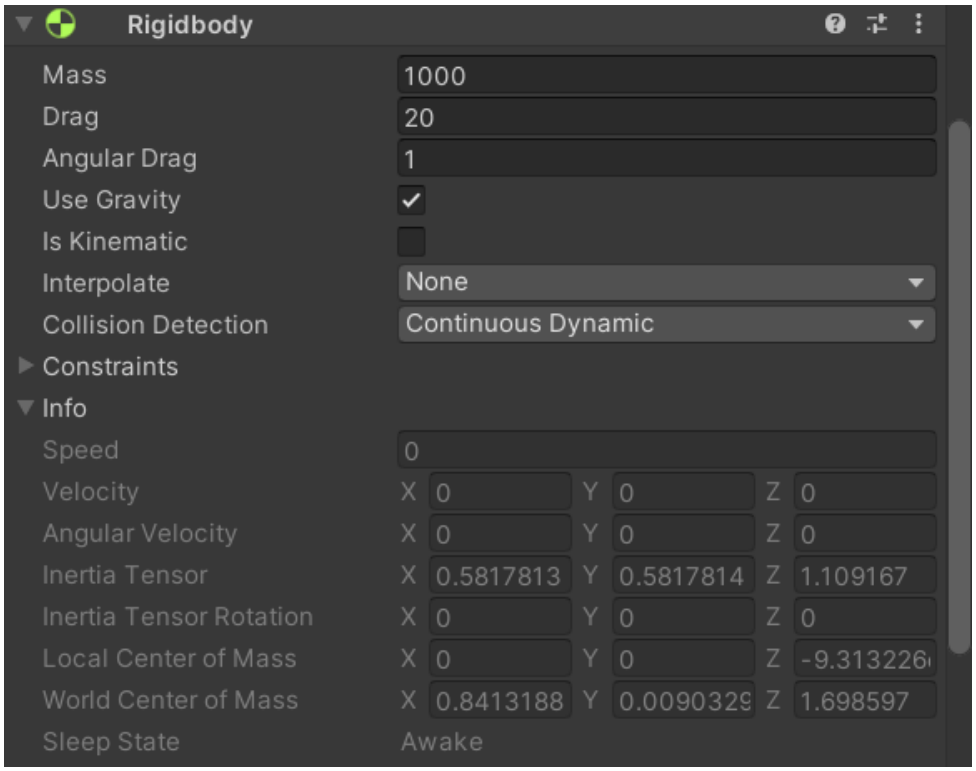


Hình 3.11: Collider của quân cờ (viền màu xanh lá)

Ban đầu nhóm chúng em định sử dụng collider dạng lưới cho các quân cờ vì đó là collider đem lại độ chính xác cao nhất. Tuy nhiên, trong quá trình làm chúng em nhận thấy việc này sẽ làm cho khối lượng tính toán của ứng dụng tăng đáng kể vì mỗi collider này sẽ bao gồm rất nhiều mặt tam giác. Bản thân việc va chạm của các quân cờ cũng không cần phải có độ chính xác cao nên bọn em quyết định thay bằng collider hình hộp chữ nhật để cải thiện hiệu suất. Bàn cờ sẽ có một collider để chặn không cho các quân cờ rơi xuống, vòng tròn gợi ý cũng có collider để người dùng có thể tương tác với chúng khi di chuyển các quân cờ bằng tay.

Mô phỏng vật lý

Unity đã tích hợp sẵn một engine mô phỏng vật lý trong môi trường 3D, các lực được mô phỏng bao gồm cả trọng lực, lực ma sát, lực cản không khi và rất nhiều các lực khác để đảm bảo việc di chuyển trong không gian của các vật thể có độ chân thực cao nhất [8]. Trong trò chơi cờ tướng bình thường thì việc mô phỏng vật lý là không cần thiết, tuy nhiên khi phát triển cho cả môi trường VR thì việc này lại cần thiết vì mình có khả năng cầm, nắm các vật thể. Để gán các tính chất vật lý lên một vật thể, Unity có sử dụng một thành phần gọi là RigidBody.



Hình 3.12: RigidBody

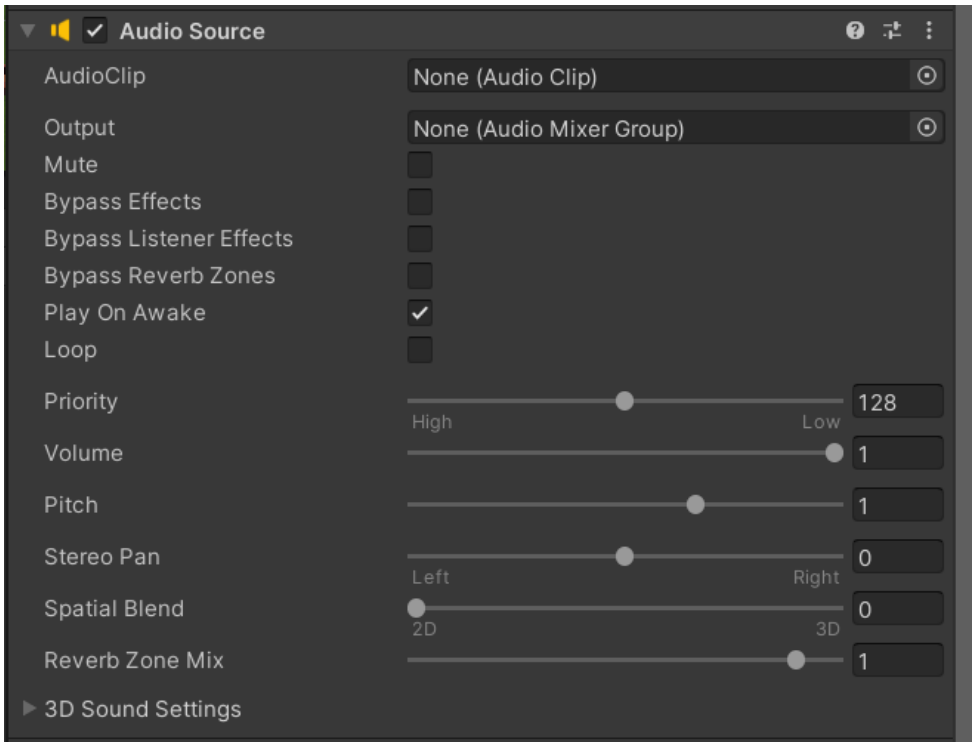
Trong quá trình làm, nhóm chúng em gặp không ít khó khăn trong việc tinh chỉnh các chỉ số của thành phần này. Chủ yếu đến từ việc khi mới khởi tạo, các quân cờ sẽ không ở ngay sát mặt bàn cờ nên chúng sẽ rơi, khi đó lực cản không khi và lực ma sát sẽ làm cho chúng rơi với quỹ đạo rất bất thường và gây lộn xộn bàn cờ.

3.4.2 Hiệu ứng âm thanh và particle

Để cho trò chơi thêm phần sống động thì việc thêm các hiệu ứng là rất cần thiết, hai loại hiệu ứng cơ bản nhất trong Unity mà nhóm chúng em có sử dụng là hiệu ứng âm thanh và hiệu ứng particle.

Âm thanh

Trong khi đánh cờ có tiếng nhạc nền ở sau, mỗi khi đánh một quân cờ có tiếng "cạch"để tăng độ chân thực, khi đến lượt bên nào trò chơi sẽ thông báo qua việc đọc tên bên đánh,... tất cả những hiệu ứng trên đều có thể dễ dàng cài đặt bằng thành phần Audio Source trong Unity.

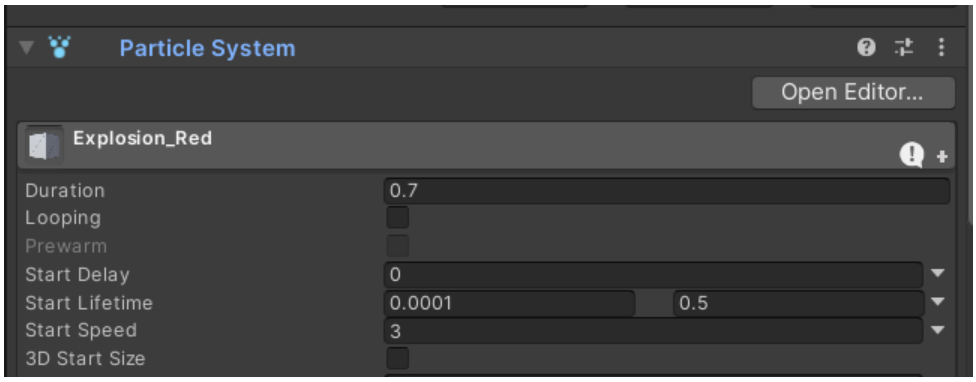


Hình 3.13: Audio Source

Có thể thấy thành phần có rất nhiều thông số để tinh chỉnh cho phù hợp với mục đích sử dụng, ngoài thông số hiển nhiên cần có là đoạn âm thanh được sử dụng, ta còn có thể cho nó chạy tự động khi trò chơi bắt đầu, cho chạy lặp, điều chỉnh cách nó phát tán trong không gian từ hai chiều thành ba chiều,... Với trò chơi này, nhóm chúng em chỉ cần sử dụng hai thông số là Play On Awake và Loop. Với nhạc nền thì ta cần cho nó chạy lặp liên tục và chạy khi trò chơi bắt đầu, vậy nên sẽ bật Loop và Play On Awake. Còn các hiệu ứng âm thanh khác chỉ bật khi thỏa mãn điều kiện nào đó nên Play On Awake sẽ tắt và ta phải tự chạy âm thanh trong code bằng hàm Play() [8].

Particle

Hiệu ứng Particle có thể hiểu đơn giản như là một hiệu ứng hình ảnh trong đó ta sẽ tạo ra các hạt (particle) liên tục. Ví dụ như trong trò chơi này, hiệu ứng Particle được sử dụng để khi một quân cờ bị ăn, nó sẽ vỡ thành trăm mảnh. Ta có thể cài đặt tính năng này bằng thành phần Particle System trong Unity, nó cũng khá tương tự như hiệu ứng âm thanh ở trên, có thể kích hoạt trong code bằng hàm Play() [8].

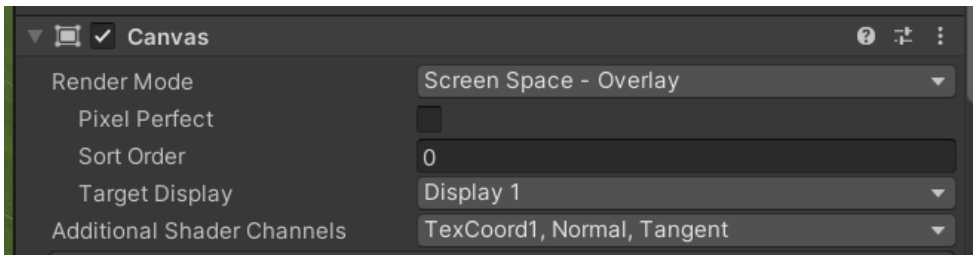


Hình 3.14: Particle System

3.4.3 Giao diện người dùng

Giao diện người dùng của trò chơi tại thời điểm hiện tại tương đối đơn giản. Ở giao diện chính của trò chơi, ngoài bàn cờ ra, có dòng chữ cho mình biết lượt hiện tại là lượt bên nào ở góc trên bên trái, bên dưới nó sẽ là hai dòng chữ khác cho tính năng nhận diện giọng nói. Dòng trên sẽ là lệnh mà trò chơi nhận diện được khi người chơi nói dưới dạng văn bản, dòng dưới cũng là lệnh đó sau khi đã đưa qua phần xử lý cú pháp lệnh. Mỗi khi có con tướng bên nào bị chiếu, ở giữa trên cùng của màn hình sẽ có dòng chữ thông báo, tương tự khi có bên nào thắng cuộc. Ở góc trên bên phải sẽ có một nút "lưu và thoát"để phục vụ cho tính năng lưu trạng thái của trò chơi.

Giao diện màn hình trước khi bắt đầu trò chơi cũng chỉ là một ảnh nền với ba nút bắt đầu, tiếp tục và thoát. Khi ấn vào nút tiếp tục, trò chơi sẽ tải lại trạng thái gần nhất của trò chơi khi người dùng lưu và thoát. Còn với nút bắt đầu, trò chơi sẽ khởi tạo với trạng thái hoàn toàn mới. Tất cả các tính năng có thể cài đặt bằng Canvas của Unity.



Hình 3.15: Canvas

3.4.4 VR

Locomotion

Đây là một trong các yếu tố quan trọng nhất của VR cho phép người dùng có thể chuyển động được trong thế giới ảo. Locomotion trong Unity được hỗ trợ dưới dạng các script - code snippet mà mình có thể gán vào các vật thể để định nghĩa hành vi của chúng. Một trong các script cơ bản nhất mà mọi dự án VR phải có là XR Origin, về cơ bản nó đại diện cho trung tâm của thế giới ảo trong môi trường VR, thường là người chơi [8].

Hình 3.16: XR Origin

Có hai thông số cần quan tâm trong script này đó là Camera GameObject và Camera Floor Offset Object. Camera GameObject là một camera đại diện cho góc nhìn của người chơi, mọi hoạt động di chuyển trong 6 bậc tự do của người chơi sẽ được cập nhật qua camera này. Camera Floor Offset Object sẽ là độ cao so với sàn của camera, có thể coi như độ cao của nhân vật trong thế giới ảo. Với script này, người chơi đã có thể quay đầu để ngắm nhìn thế giới xung quanh và nhìn bàn cờ với quân cờ. Nhưng để tương tác được với chúng bằng tay cầm thì cần thêm vài thành phần nữa, cụ thể là XR Controller và XR Interaction Manager.

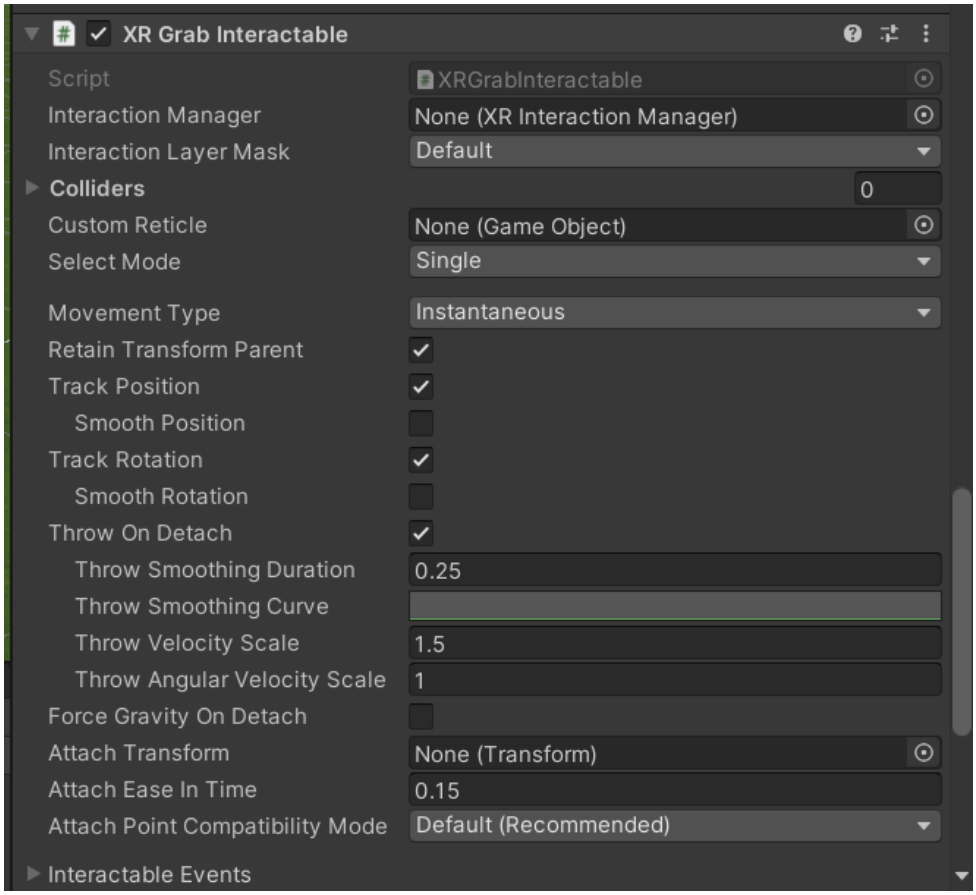
Hình 3.17: Cấu trúc một cây XR cơ bản cho locomotion

Chúng ta sẽ có hai vật thể, RightHand Controller và LeftHand Controller, đại diện cho tay cầm bên phải và bên trái tương ứng. Mỗi vật thể sẽ được gán script XR Controller, script này sẽ có vai trò theo dõi chuyển động tịnh tiến và quay của tay cầm để cập nhật cho vật thể được gán script. Ngoài ra nó còn được sử dụng để nhận các input của người dùng.

XR Interaction Manager đóng vai trò như một trung gian giữa vật tương tác và vật bị tương tác trong môi trường VR. Ví dụ như trong trò chơi này, vật tương tác sẽ là hai tay cầm còn vật bị tương tác sẽ là các quân cờ. Input của tay cầm có thể được đọc bởi XR Controller, nhưng để nó truyền đến được các quân cờ để thực hiện lệnh thì buộc phải đi qua XR Interaction Manager. Mỗi dự án cần ít nhất một XR Interaction Manager để người chơi có thể tương tác được với môi trường [8].

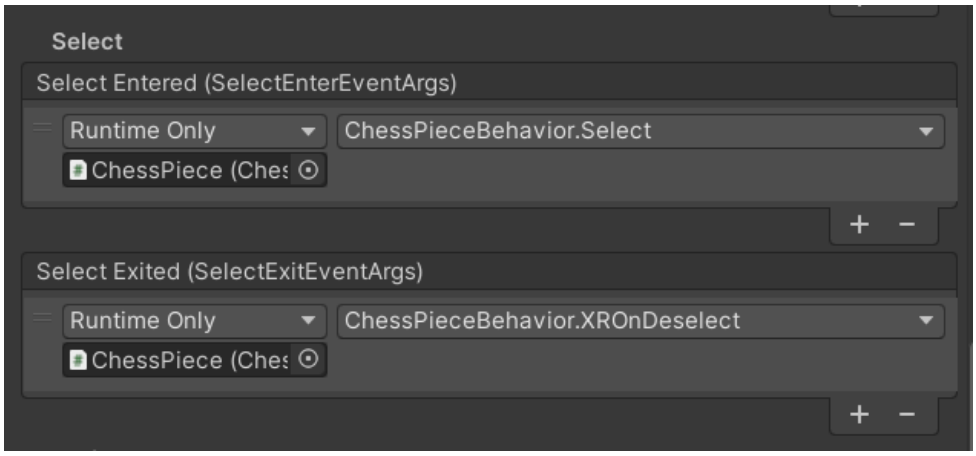
Vật thể cầm, nắm được

Một trong các tương tác cơ bản nhất trong môi trường VR là người chơi có thể cầm, nắm được các vật thể xung quanh. Điều này có thể đạt được bằng cách gán cho vật thể script XR Grab Interactable [8].



Hình 3.18: XR Grab Interactable

Hầu hết các thông số mặc định trong script này đã cho phép khả năng cầm, nắm cơ bản. Nếu muốn vật thể có hành vi đặc biệt trong một số hoạt động, ví dụ như khi mình chạm vào vật, cầm vật hay thả vật ra thì ta sẽ định nghĩa trong Interactable Events. Trong trò chơi này, khi mình cầm một quân cờ, trò chơi sẽ hiện ra các nước đi gợi ý thông qua các vòng tròn ở vị trí mà quân cờ có thể đi được. Khi thả tay ra, nếu mình thả ở một trong các vòng tròn thì quân cờ sẽ di chuyển đến chỗ đó, còn không thì nó sẽ quay trở lại chỗ cũ. Hành vi này hoàn toàn có thể cài đặt thông qua Interaction Events.



Hình 3.19: Định nghĩa hành vi khi người chơi cầm, thả quân cờ

Tương tác bằng tia

Khi tương tác bằng tay cầm, một vấn đề cần quan tâm là người dùng thực sự di chuyển nó trong không gian thực, dẫn đến việc có thể bị va chạm với các vật cản. Vậy nên mặc dù việc có thể đưa tay xuống chạm trực tiếp vào các quân cờ để di chuyển chúng sẽ tăng tính chân thực của trò chơi, nó có thể gây bất tiện cho người dùng vì họ không thể kiểm soát được môi trường xung quanh khi đang đeo kính thực tế ảo. Do đó, nhóm chúng em đề ra một cách đó là tương tác với các vật thể từ xa bằng tia. Về cơ bản, từ tay cầm của người dùng trong thế giới ảo sẽ bắn ra một tia và tia này chỉ vào vật thể nào, hay chính xác hơn là tia chạm vào collider của vật nào thì người dùng sẽ tương tác với vật thể đó.

Hình 3.20: Tương tác bằng tia

Phương án này có thể được cài đặt bằng cách gán Script XR Ray Interactor cho vật thể đại diện cho tay cầm. Ngoài ra ta cũng cần gán cho nó một thành phần có thể vẽ được tia (renderer) để người dùng có thể thấy rõ được tay mình đang chỉ vào đâu. Khi tương tác với vật thể từ xa như thế này, mọi hành vi được định nghĩa trong Interactable Events của vật thể vẫn sẽ được giữ nguyên, đảm bảo logic của trò chơi. Tuy nhiên, một vấn đề được nảy sinh đó là tia bắn ra từ tay cầm người dùng sẽ chỉ vào rất nhiều vật thể và ta cần phải có cách phân biệt giữa chúng, điều này có thể dễ dàng thực hiện được thông qua layer mask. Ta có thể hiểu nó như một cách để phân nhóm các vật thể. Mỗi vật thể trong Unity sẽ thuộc về một layer, dựa vào layer này ta có thể cho phép vật thể nào có thể tương tác được với nhau.

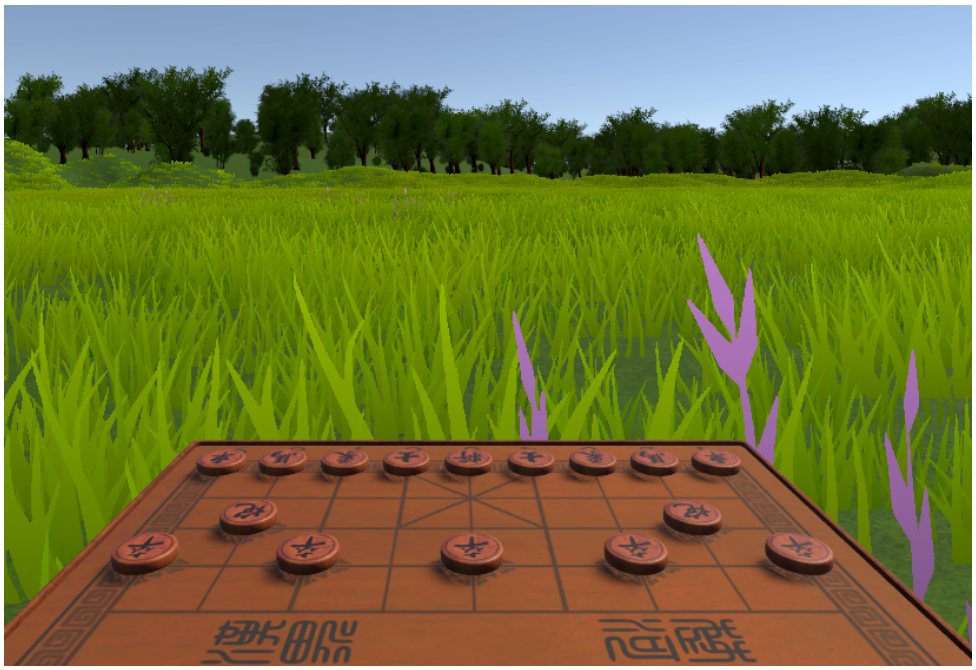
Nhóm chúng em cũng đề ra một thiết kế là chỉ cho phép người dùng thấy một tay tại một thời điểm. Thiết kế này ban đầu được đưa ra để ngăn không cho người chơi nhặt lên hai quân cờ cùng một lúc. Mặc dù có một cách khác là đếm số quân cờ đang được tương tác và chỉ cho phép có một quân cờ như vậy nhưng khi đó ta vẫn sẽ phải xử lý các luồng sự kiện được bắn ra từ hai tay cùng một lúc. Việc này có thể làm cho mã nguồn phức tạp hơn cần thiết. Trong thực tế khi chơi cờ tướng, người chơi cũng chỉ dùng một tay để nhặt các quân cờ, vậy nên chỉ cho hiển thị một tay không phải là thiết kế gì đó quá khác thường.

4 Kết quả

Với chuẩn cờ tướng nhóm chúng em và nhóm AI cờ tướng xây dựng nên, cùng với mô hình chuyển đổi giọng nói cờ tướng sang văn bản, nhóm chúng em đã hoàn thành việc xây dựng game cờ tướng với đầy đủ các chức năng: VR, AI, logic game, môi trường trong game.

Về logic game, game đã có đầy đủ các cơ chế cần thiết của một game cờ tướng cơ bản. Người dùng có thể chơi một ván cờ đến trạng thái kết thúc (thắng hoặc thua). Người dùng cũng có thể lưu lại trạng thái của bàn cờ để có thể tiếp tục chơi tại một thời điểm khác trong tương lai.

Với chức năng thực tế ảo, game đã cho phép người chơi sử dụng tay cầm để tương tác với quân cờ, sử dụng cơ chế dò tia để chọn quân. Ngoài ra game cũng cho phép người dùng di chuyển và nhìn được môi trường xung quanh toàn cảnh 360°.



Hình 4.1: Môi trường trong game cờ tướng, với góc nhìn VR

Với chức năng AI, người dùng có thể điều khiển quân cờ theo tập lệnh mà 2 nhóm chúng em xây dựng nên. Để đánh giá độ chính xác của mô hình nhận diện giọng nói cho game cờ tướng, nhóm AI cờ tướng có chạy so sánh mô hình wav2vec2 gốc (BASE) và mô hình wav2vec2 gốc kết hợp với mô hình ngôn ngữ được xây trên bộ dữ liệu cờ tướng và beam search (OUR). Để được những đoạn âm thanh để đánh giá, nhóm AI sử dụng dịch vụ text-to-speech của fpt.ai và đồng thời để tăng độ khó cho mô hình, nhóm sử dụng giọng của miền Trung và miền Nam, nhưng đoạn văn bản dùng để sinh sẽ được lấy ngẫu nhiên 1000 lệnh cờ tướng theo chuẩn của nhóm đề ra (bộ dữ liệu: shorturl.at/fglru). Và kết quả như trong bảng 1, mô hình OUR dự đoán đúng vượt trội so với mô hình BASE gốc.

Model	Accuracy
BASE	7.5%
OUR	78.8%

Bảng 1: Hiệu suất của các mô hình thử nghiệm

Về tốc độ xử lý của mô hình, thời gian phản hồi trung bình với 20 lệnh cờ tướng chúng em gửi đến model từ game, được tính tính bằng đồng hồ của Unity, gửi tới server là localhost, từ thời điểm lệnh bắt đầu được gửi cho tới khi lệnh được thực thi trong game là **3.3 giây**.

Tài liệu tham khảo

[1] Alexander Veysov **and** Dimitrii Voronin. "One Voice Detector to Rule Them All". **in** *The Gradient*: (2022).

[2] *4 use cases for virtual reality in the military and defense industry*. URL: http://blog.techviz.net/4-use-cases-for-virtual-reality-in-the-military-and-defense-industry.

[3] Cloud Hu. *ChineseChessVR*. URL: https://github.com/cloudhu/ChineseChessVR.

[4] *Junior Programmer*. URL: https://learn.unity.com/pathway/junior-programmer.

[5] Steven M. LaValle. "12". **in** *VIRTUAL REALITY*.

[6] Paul Sawers. *Medical simulation platform FundamentalVR raises 20M$ to help surgeons learn through VR*. URL: https://techcrunch.com/2022/08/11/medical-simulation-platform-fundamentalvr-raises-20m-to-help-surgeons-learn-through-vr.

[7] *Unity Essentials*. URL: https://learn.unity.com/pathway/unity-essentials.

[8] *Unity Manual*. URL: https://docs.unity3d.com/Manual/ScriptingSection.html.

[9] *What can Unity do?* URL: https://learn.unity.com/tutorial/what-can-unity-do?.

[10] *What is virtual reality in travel?* URL: https://immersionvr.co.uk/about-360vr/vr-for-tourism/.