HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

ĐỒ ÁN
TỐT NGHIỆP ĐẠI HỌC

Đề tài: **"Xây dựng trò chơi ứng dụng thiết bị nhận diện cử chỉ bàn tay Leap Motion Controller trên Unity Engine"**

Giảng viên hướng dẫn	**: ThS. BÙI VĂN KIÊN**
Sinh viên thực hiện	**: NGUYỄN TIẾN HÙNG**
Lớp	**: D20CNPM05**
Mã sinh viên	**: B20DCCN297**
Khóa	**: 2020-2025**
Hệ đào tạo	**: ĐH CHÍNH QUY**

Hà Nội – 2024

NHẬN XÉT, ĐÁNH GIÁ, CHO ĐIỂM

(Của giảng viên hướng dẫn)

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

Điểm: ……..……(bằng chữ: …..…….……)

Đồng ý/Không đồng ý cho sinh viên bảo vệ trước hội đồng chấm đồ án tốt nghiệp?

Hà Nội, ngày tháng năm 20

CÁN BỘ - GIẢNG VIÊN HƯỚNG DẪN

(ký, họ tên)

NHẬN XÉT, ĐÁNH GIÁ, CHO ĐIỂM

(Của giảng viên phản biện)

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

Điểm: ……..……(bằng chữ: …..…….……)

Đồng ý/Không đồng ý cho sinh viên bảo vệ trước hội đồng chấm đồ án tốt nghiệp?

Hà Nội, ngày tháng năm 20

CÁN BỘ - GIẢNG VIÊN PHẢN BIỆN

(ký, họ tên)

LỜI CẢM ƠN

Lời đầu tiên em xin gửi lời cảm ơn đến ThS. Bùi Văn Kiên, người đã tận tình hướng dẫn và đồng hành cùng em trong suốt quá trình thực hiện đồ án tốt nghiệp. Nhờ sự góp ý, chỉ dẫn nhiệt tình của thầy, em đã có thể triển khai và hoàn thiện tốt nhất các nội dung trong đề tài của mình.

Tiếp theo, em xin gửi lời cảm ơn chân thành tới ThS. Nguyễn Đức Hoàng, người đã định hướng đề tài, hỗ trợ thiết bị cần thiết và kết nối tới các anh chị có kinh nghiệm giúp em có được điều kiện thuận lợi nhất để thực hiện đồ án. Sự hỗ trợ của thầy đã giúp em vượt qua nhiều khó khăn trong quá trình nghiên cứu và thực hiện đề tài.

Ngoài ra, em cũng xin bày tỏ lòng biết ơn tới các thầy cô giáo trong Học viện Công nghệ Bưu chính Viễn thông, đặc biệt là các thầy cô khoa Công nghệ thông tin I, đã luôn tận tụy giảng dạy, truyền đạt kiến thức và kỹ năng cần thiết để em hoàn thành đồ án tốt nghiệp cũng như chuẩn bị hành trang cho công việc tương lai.

Cuối cùng, em xin gửi lời cảm ơn sâu sắc đến gia đình và bạn bè đã luôn động viên, ủng hộ và là nguồn động lực lớn lao cho em trong suốt thời gian thực hiện đồ án.

Trong quá trình thực hiện, dù đã cố gắng hết sức nhưng chắc chắn không tránh khỏi những thiếu sót. Em rất mong nhận được sự góp ý của thầy cô và các bạn để đề tài của em được hoàn thiện hơn.

Em xin chân thành cảm ơn!

Hà Nội, ngày 28 tháng 12 năm 2024

Người thực hiện

Nguyễn Tiến Hùng

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC CÁC TỪ VIẾT TẮT

AI: Artificial Intelligence	Trí tuệ nhân tạo
VR: Virtual Reality	Thực tế ảo
AR: Augmented Reality	Thực tế tăng cường
XR:Extended Reality	Thực tế mở rộng
3D: Three Dimensions	Không gian 3 chiều
USB: Universal Serial Bus	Cổng kết nối đa năng
IoT: Internet of Things	Vạn vật kết nối Internet
SDK: Software Development Kit	Bộ công cụ phát triển phần mềm

LỜI MỞ ĐẦU

Trong bối cảnh xã hội tiến bộ ngày nay khi công nghệ được phát triển không ngừng, mỗi chúng ta hẳn không còn xa lạ gì với khái niệm công nghệ thực tế ảo VR-AR. Không chỉ mang lại những trải nghiệm mới mẻ mà nó còn mở ra nhiều tiềm năng ứng dụng đa dạng trong các lĩnh vực như giáo dục, y tế, giải trí, và đặc biệt là trò chơi điện tử. Leap Motion là thiết bị cảm biến chuyển động tiên tiến, cho phép người dùng tương tác với máy tính thông qua cử chỉ tay, mang đến sự kết hợp hài hòa giữa công nghệ hiện đại và tính trực quan trong thiết kế tương tác.

Đồ án tập trung khai thác tiềm năng của công nghệ này trong lĩnh vực giải trí chơi, với mục tiêu phát triển game Chém hoa quả (Fruit Ninja) – 1 tựa game thành công và nổi tiếng toàn cầu, hẳn ai cũng đã từng chơi hoặc xem và hình dung ra được lối chơi của trò chơi này trên các thiết bị cảm ứng nhưng giờ đây nó sẽ được trải nghiệm theo 1 cách hoàn toàn khác.

Nội dung đồ án gồm các phần như sau:

Chương 1: Tổng quan công nghệ nhận diện cử chỉ tay và thiết bị Leap Motion Controller

Chương 2: Giới thiệu công cụ phát triển game Unity Engine và plugin Ultraleap

Chương 3: Phân tích và thiết kế game Chém hoa quả

Chương 4: Xây dựng game Chém hoa quả tích hợp Leap Motion trên Unity

CHƯƠNG 1. TỔNG QUAN CÔNG NGHỆ NHÂN DIỆN CỬ CHỈ BÀN TAY VÀ LEAP MOTION CONTROLLER

1.1. Công nghệ nhận diện cử chỉ bàn tay

1.1.1. Bài toán chung

Bài toán nhận diện cử chỉ tay đã được quan tâm và nghiên cứu từ những năm đầu thế kỷ trước. Mục tiêu là nhận diện được hình dáng (cử chỉ tay tĩnh) hoặc hành động (cử chỉ tay động) của cử chỉ tay, bao gồm ngón tay, bàn tay, cánh tay để từ đó đưa ra các thông tin hữu ích trong tương tác người – máy.

Trước đây bài toán nhận dạng cử chỉ tay thường được tiếp cận theo hướng áp dụng thị giác máy tính. Theo đó các cử chỉ ngón tay, bàn tay sẽ được chụp lại và sử dụng các mô hình học máy để huấn luyện và nhận dạng. Cách tiếp cận này đạt được độ chính xác cao trong nhận dạng các cử chỉ tay tĩnh nhờ khả năng xử lý tốt của các thuật toán máy học. Tuy nhiên, nó có những hạn chế nhất định, đặc biệt khi áp dụng vào nhận diện các cử chỉ tay động. Phạm vi nhận dạng bị giới hạn trong vùng nhìn của camera, và hệ thống thường đòi hỏi tài nguyên tính toán lớn, khiến nó không phù hợp với các ứng dụng yêu cầu độ phản hồi thời gian thực cao hoặc hoạt động trong môi trường đa dạng.

Với sự phát triển mạnh mẽ của công nghệ vi điện tử, đặc biệt trong thập kỷ gần đây đã mở ra những hướng tiếp cận mới cho bài toán nhận diện cử chỉ tay. Các thiết bị cảm biến đã vượt qua những giới hạn của thị giác máy tính truyền thống bằng cách sử dụng dữ liệu không gian ba chiều (3D) để phân tích chính xác các cử động của bàn tay và ngón tay. Công nghệ này không chỉ tăng cường độ chính xác trong nhận diện mà còn mở rộng phạm vi ứng dụng, cho phép người dùng thực hiện các thao tác ở nhiều vị trí và điều kiện môi trường khác nhau mà không cần camera trực diện. Kết hợp với các thuật toán tiên tiến như học sâu (Deep Learning), hệ thống hiện nay có thể không chỉ nhận diện cử chỉ tay tĩnh mà còn phân tích được các chuỗi hành động phức tạp theo thời gian.

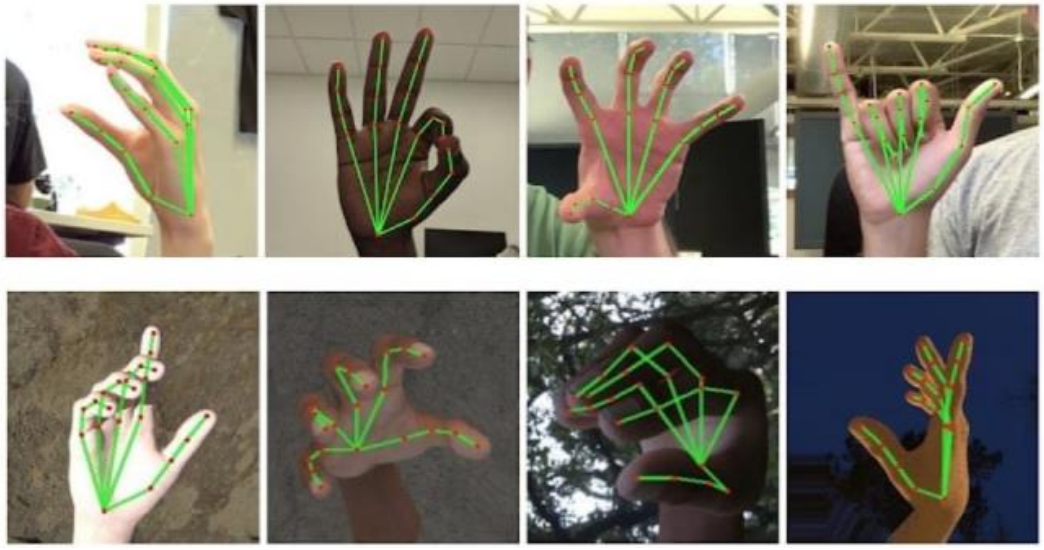
1.1.2. Nguyên lý hoạt động

Nguyên lý hoạt động của các thiết bị nhân diện cử chỉ bàn tay dựa trên việc thu thập dữ liệu từ bàn tay (vị trí, hình dạng, và chuyển động) sau đó phân tích để nhận diện các cử chỉ hoặc hành vi mong muốn.

Đối với các thiết bị dựa trên thị giác máy tính (Computer Vision-Based Systems), nó hoạt động bằng cách sử dụng một hoặc nhiều camera để ghi nhận hình ảnh hoặc video của bàn tay trong thời gian thực. Quá trình xử lý diễn ra qua các bước:

1. **Ghi nhận hình ảnh**: Camera thu thập hình ảnh của bàn tay. Một số thiết bị cao cấp sử dụng camera chiều sâu (depth camera), cho phép phân biệt giữa các đối tượng trong không gian ba chiều và cải thiện độ chính xác.
2. **Xử lý hình ảnh**: Các thuật toán xử lý ảnh sẽ tách biệt bàn tay ra khỏi nền bằng các kỹ thuật như phân đoạn hình ảnh (image segmentation) hoặc phát hiện biên (edge detection). Sau đó, bàn tay được trích xuất dưới dạng hình dạng 2D hoặc mô hình 3D.
3. **Nhận diện cử chỉ**: Dựa trên hình dạng hoặc chuyển động của bàn tay, các hệ thống sử dụng các mô hình học máy hoặc học sâu để nhận diện các cử chỉ tĩnh (như giơ tay, chỉ ngón) hoặc động (như vẫy tay, xoay cổ tay).

Hệ thống này phù hợp cho nhiều ứng dụng, nhưng hiệu quả có thể bị ảnh hưởng bởi môi trường ánh sáng hoặc sự che khuất của bàn tay



Hình 1.1. Nhận diện cử chỉ tay với thị giác máy tính

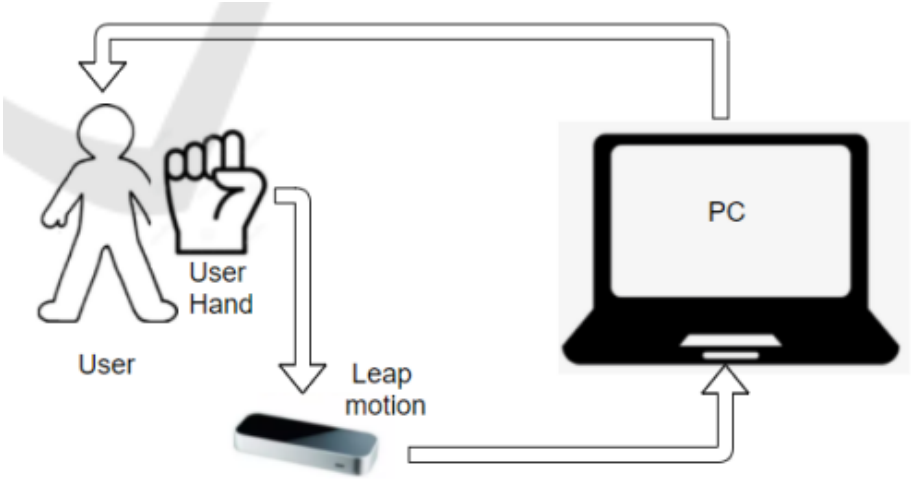
Đối với nhóm thiết bị sử dụng cảm biến chuyên dụng (Sensor-Based Systems):

1. **Thu thập dữ liệu từ cảm biến**:
   - Cảm biến hồng ngoại (Infrared Sensors): Thiết bị phát ra ánh sáng hồng ngoại và ghi nhận tín hiệu phản xạ từ bề mặt bàn tay để xác định vị trí, hình dạng, và độ sâu trong không gian 3D.

- Cảm biến siêu âm (Ultrasonic Sensors): Đo thời gian sóng âm phản xạ để xác định khoảng cách và hướng di chuyển..
- Cảm biến điện tử gắn trên tay (Wearable Sensors): Ghi nhận chuyển động của các khớp và độ nghiêng từ gia tốc kế, con quay hồi chuyển hoặc cảm biến lực.

2. **Tái tạo mô hình bàn tay**: Sử dụng dữ liệu để xây dựng mô hình xương và khớp bàn tay trong không gian 3D (skeleton model).

Nhóm thiết bị này có ưu điểm là hoạt động ổn định hơn trong nhiều môi trường và cho phép nhận diện cử chỉ tay động một cách chính xác hơn.



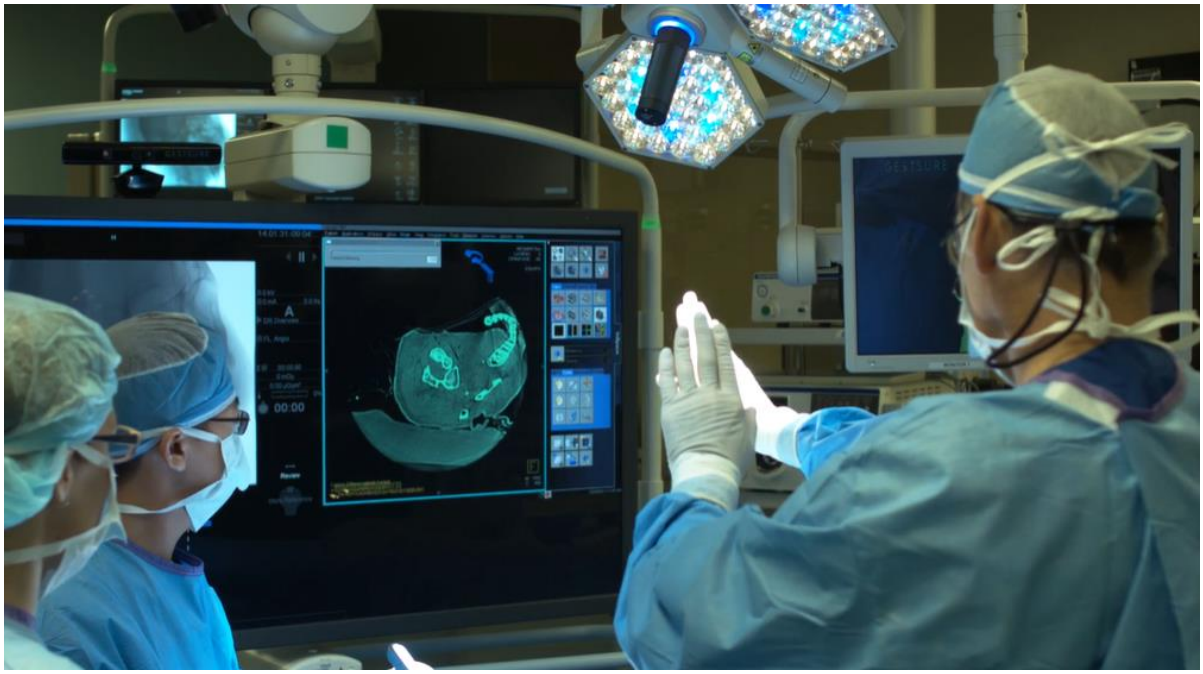
Hình 1.2. Sơ đồ hoạt động của thiết bị cảm biến

1.1.3. Khả năng ứng dụng

Công nghệ nhận diện cử chỉ bàn tay đang dần trở thành một phần không thể thiếu trong nhiều lĩnh vực khác nhau nhờ tính tương tác tự nhiên và không cần chạm.

Nó được sử dụng rộng rãi trong các thiết bị VR/AR để tăng cường trải nghiệm tương tác, giúp người dùng tương tác một cách tự nhiên với thế giới ảo mà không cần sử dụng tay cầm hay các công cụ ngoại vi khác. Một số sản phẩm VR/AR nổi bật có sử dụng công nghệ này như: Oculus Quest của Meta, Apple Vision Pro hay HoloLens của Microsoft, …

Trong y tế, việc sử dụng cử chỉ tay giúp các bác sĩ và nhân viên y tế tương tác với hệ thống mà không cần tiếp xúc vật lý, đặc biệt trong các phòng mổ, nơi yêu cầu vệ sinh cao. Có thể kể đến như GestSure đang tiên phong trong việc ứng dụng công nghệ này để phép bác sĩ điều khiển hình ảnh y tế trong khi phẫu thuật mà không cần chạm vào màn hình, giúp giảm thiểu nguy cơ nhiễm trùng và nâng cao hiệu quả công việc.



Hình 1.3. Ứng dụng nhận diện cử chỉ bàn tay trong y tế

Trong ngành công nghiệp game, sự phát triển của các hệ thống tương tác không chạm đã tạo ra những trải nghiệm chơi game mới mẻ, trong đó người chơi có thể trực tiếp tham gia vào các hành động trong game thông qua cử chỉ tay.

Công nghệ nhận diện cử chỉ tay cũng đã được ứng dụng trong giáo dục, đặc biệt là trong các lớp học ảo và mô phỏng, nơi học sinh và giáo viên có thể tương tác với các mô hình 3D mà không cần sử dụng thiết bị ngoại vi, giúp tăng tính trực quan và sinh động cho quá trình học, thậm chí giúp học sinh dễ dàng tiếp cận các khái niệm phức tạp.

Trong lĩnh vực sản xuất và công nghiệp, các công ty lớn đã bắt đầu ứng dụng cử chỉ tay để điều khiển robot và dây chuyền sản xuất mà không cần tiếp xúc vật lý, giúp tăng cường hiệu quả công việc và giảm thiểu rủi ro.

Công nghệ nhận diện cử chỉ tay cũng đã được ứng dụng trong các thiết bị nhà thông minh, nơi người dùng có thể điều khiển các thiết bị trong nhà mà không cần sử dụng remote hay công tắc.

Với tiềm năng phát triển lớn, công nghệ này đang hướng đến việc kết hợp với trí tuệ nhân tạo (AI) để tạo ra các hệ thống tương tác thông minh hơn, có khả năng học hỏi và thích nghi với hành vi của từng cá nhân. Trong tương lai, các thiết bị nhận diện cử chỉ tay có thể được tích hợp sâu hơn vào các thiết bị di động, wearable, hoặc các thiết bị IoT, mở

ra khả năng tương tác linh hoạt hơn và thúc đẩy sự phát triển của môi trường số hóa toàn diện. Những tiến bộ này hứa hẹn sẽ đưa công nghệ nhận diện cử chỉ bàn tay từ một công cụ hỗ trợ trở thành một thành phần cốt lõi trong tương tác giữa con người và máy móc.

1.2. Leap Motion Controller

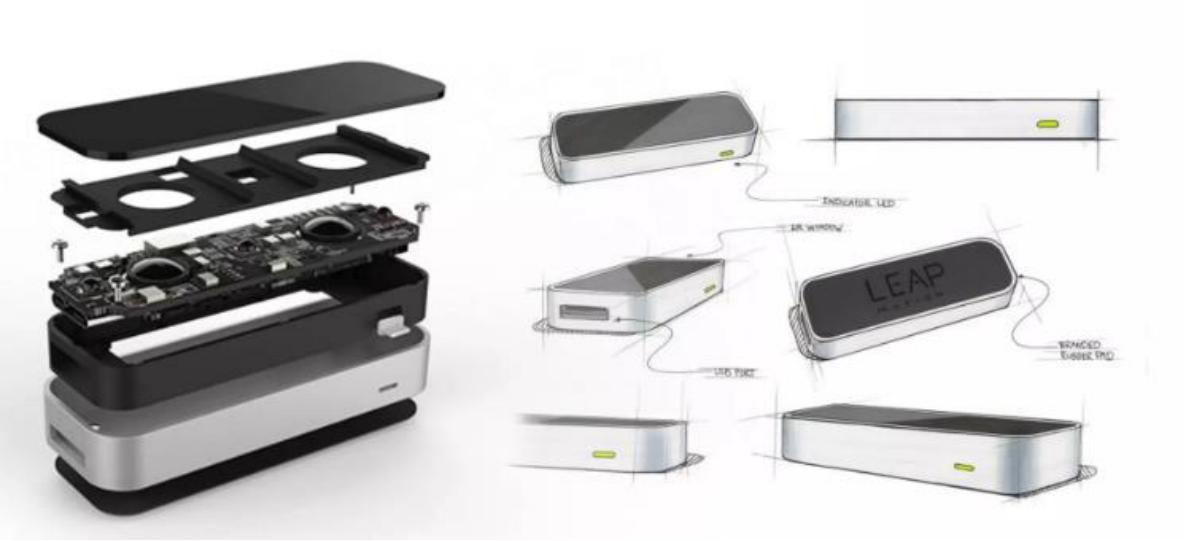
1.2.1. Giới thiệu thiết bị

Leap Motion Controller là một thiết bị ngoại vi được thiết kế để theo dõi và nhận diện các chuyển động tay và ngón tay của người dùng trong không gian ba chiều. Thiết bị này được phát triển bởi Leap Motion Inc [1], một công ty công nghệ có trụ sở tại San Francisco, Mỹ. Với kích thước nhỏ gọn và khả năng theo dõi chính xác, Leap Motion Controller đã mở ra một hướng đi mới trong cách con người tương tác với máy tính và các hệ thống kỹ thuật số mà không cần sử dụng đến chuột, bàn phím hay các thiết bị điều khiển truyền thống.

Hình 1.4. Thiết bị Leap Motion Controller

Thiết bị nhỏ gọn và nhẹ, mang thiết kế hiện đại với vẻ ngoài cao cấp. Khung viền được chế tạo từ nhôm nguyên khối, các góc bo tròn tinh tế, và mặt trên là một tấm kính tối màu trong suốt, giúp thiết bị theo dõi chính xác các chuyển động tay trong không gian. Phần kính này bảo vệ hai cảm biến và đèn LED hồng ngoại, hỗ trợ thiết bị hoạt động bền bỉ và hiệu quả. Mặt đế được trang bị lớp cao su chống trượt với logo Leap Motion khắc

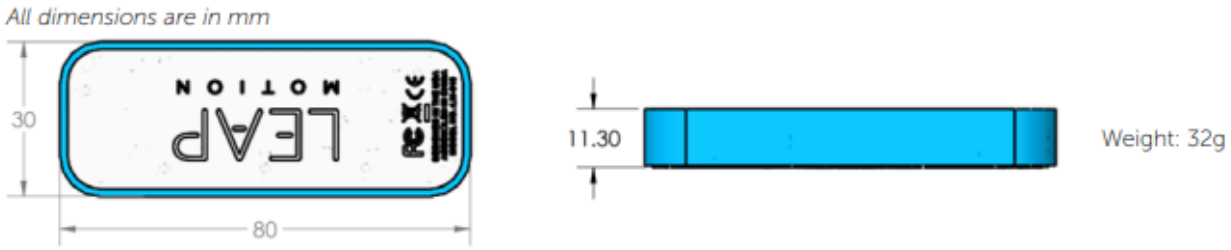
chìm, đảm bảo sự ổn định khi sử dụng. Thiết bị được kết nối với máy tính qua cổng mini USB, đi kèm đèn LED hiển thị trạng thái hoạt động [2].



Hình 1.5. Cấu trúc Leap Motion Controller

1.2.2. Cấu tạo và nguyên lý hoạt động

Leap Motion Controller có thiết kế nhỏ gọn với kích thước chỉ 80 x 30 x 11.3 mm và trọng lượng rất nhẹ chỉ 32g [3].

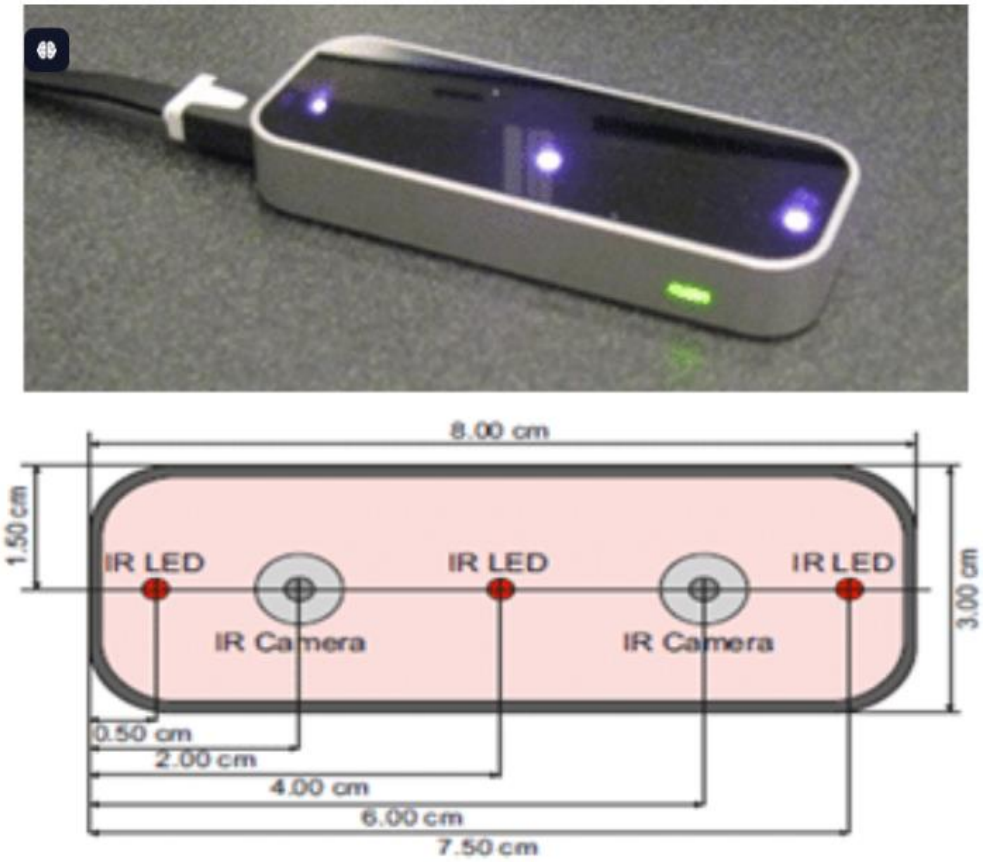


Hình 1.6. Kích thước và trọng lượng Leap Motion Controller

Thiết bị bao gồm các thành phần chính sau:

- Cảm biến hồng ngoại (Infrared Sensors): Có hai cảm biến hồng ngoại nằm bên trong thiết bị, chịu trách nhiệm thu thập hình ảnh 3D của bàn tay và các vật thể nằm trong phạm vi theo dõi.
- Đèn LED hồng ngoại (Infrared LEDs): Ba đèn LED hồng ngoại được sử dụng để chiếu sáng vùng không gian mà thiết bị theo dõi. Ánh sáng hồng ngoại không gây ảnh hưởng đến mắt người nhưng lại giúp cảm biến ghi lại hình ảnh rõ ràng, ngay cả trong điều kiện ánh sáng yếu.

- Vi xử lý (Processor): Thiết bị tích hợp một vi xử lý để xử lý sơ bộ dữ liệu hình ảnh trước khi gửi tới máy tính qua cổng USB.
- Cổng kết nối USB: Kết nối giữa Leap Motion Controller và máy tính thông qua cổng USB, vừa cung cấp nguồn điện cho thiết bị, vừa truyền dữ liệu.



Hình 1.7. Cấu tạo bên trong Leap Motion Controller

Nguyên lý hoạt động của Leap Motion dựa trên việc phát và thu ánh sáng hồng ngoại để xây dựng bản đồ 3D của các vật thể trong vùng quan sát:

- Các đèn LED phát ánh sáng hồng ngoại chiếu vào bàn tay hoặc các vật thể trong phạm vi hoạt động (tối đa khoảng 60 cm).
- Cảm biến hồng ngoại ghi nhận ánh sáng phản xạ lại từ bàn tay và các vật thể này.
- Dữ liệu từ cảm biến được chuyển đổi thành hình ảnh 3D và gửi tới máy tính.
- Phần mềm Leap Motion SDK xử lý dữ liệu để xác định vị trí, hình dạng và chuyển động của tay, từ đó nhận diện các cử chỉ như kéo, thả, vuốt, hoặc phóng to/thu nhỏ.

Nhờ vào thiết kế tinh vi và các thuật toán mạnh mẽ, Leap Motion Controller đạt độ chính xác cao đến từng milimet, với khả năng nhận diện chuyển động của từng ngón tay ngay cả khi chúng di chuyển nhanh.

1.2.3. Lịch sử phát triển

Leap Motion Controller được giới thiệu lần đầu tiên vào năm 2012 và chính thức ra mắt vào năm 2013. Thiết bị này ngay lập tức gây chú ý nhờ thiết kế nhỏ gọn, hiện đại và khả năng theo dõi chuyển động tay trong không gian 3D với độ chính xác cao (độ sai lệch dưới 0,01 mm). Phiên bản đầu tiên bao gồm:

- Phần cứng: Một thiết bị nhỏ gọn với khung nhôm nguyên khối, bề mặt kính cường lực chống xước. Bên trong tích hợp hai camera hồng ngoại và ba đèn LED hồng ngoại để theo dõi chuyển động tay.
- Phần mềm: SDK ban đầu hỗ trợ các hệ điều hành phổ biến như Windows và macOS, cung cấp khả năng phát hiện ngón tay, lòng bàn tay và cử chỉ cơ bản như nhấn, vuốt, nắm, và chỉ.

Hình 1.8. Sử dụng Leap Motion Controller phiên bản đầu tiên

Vào năm 2016, Leap Motion ra mắt phiên bản phần mềm Leap Motion Controller v2, cùng với cải tiến phần cứng nhằm mở rộng khả năng theo dõi chuyển động:

- Phạm vi theo dõi tăng: Tầm hoạt động được mở rộng để theo dõi bàn tay và ngón tay ngay cả khi chúng không trực tiếp đối diện với thiết bị.

- Cải tiến phần mềm: SDK v2 hỗ trợ theo dõi chính xác hơn, bao gồm cả chuyển động phức tạp như gập ngón tay, cử động cổ tay và tương tác hai tay cùng lúc.
- Tích hợp với VR/AR: Hỗ trợ các hệ thống VR/AR thông qua phụ kiện gắn thiết bị lên kính VR như Oculus Rift và HTC Vive.

Phiên bản nâng cấp đã giải quyết nhiều hạn chế trước đó, đồng thời mở rộng ứng dụng vào thực tế ảo (VR) và thực tế tăng cường (AR), làm cho Leap Motion trở thành thiết bị phổ biến trong cộng đồng phát triển VR/AR.

Hình 1.9. Leap Motion Controller gắn trên kính VR

Năm 2019, Leap Motion sáp nhập với Ultrahaptics, một công ty công nghệ tập trung vào cảm giác xúc giác siêu âm. Sau sáp nhập, Leap Motion đổi tên thành Ultraleap, tập trung vào việc phát triển công nghệ theo dõi bàn tay và trải nghiệm xúc giác không chạm.

Cũng trong giai đoạn này, Ultraleap giới thiệu phiên bản nâng cấp của thiết bị, thường được gọi là Leap Motion Controller 2 hoặc Ultraleap Hand Tracking Module. Thiết bị này mang lại nhiều cải tiến so với phiên bản đầu tiên:

- Tăng góc nhìn: Góc theo dõi mở rộng lên tới 180 độ theo chiều ngang và 170 độ theo chiều dọc, hỗ trợ trải nghiệm tự nhiên hơn.
- Phạm vi hoạt động lớn hơn: Khoảng cách theo dõi tăng lên tới 1 mét, so với giới hạn 60 cm của phiên bản đầu.
- Độ chính xác và tốc độ cao hơn: Độ trễ thấp hơn, nhận diện chính xác hơn cả trong các chuyển động phức tạp.
- Tích hợp chuyên sâu: Thiết kế dành cho các thiết bị VR/AR như kính Varjo XR-3 hoặc hệ thống công nghiệp.

Hình 1.10. Thiết bị Leap Motion Controller 2

Sau Leap Motion Controller 2, Ultraleap tiếp tục mở rộng các dòng sản phẩm:

- Ultraleap Stereo IR 170 Module: Thiết bị theo dõi bàn tay tích hợp, tối ưu hóa cho các thiết bị VR/AR công nghiệp.
- TouchFree Application: Giải pháp không chạm dành cho màn hình cảm ứng, được ứng dụng trong y tế, bán lẻ, và các hệ thống công cộng.
- Haptics Technology: Kết hợp công nghệ theo dõi tay và cảm giác xúc giác không chạm trong không gian 3D.

1.2.4. Các ứng dụng

Leap Motion Controller đã trở thành một phần không thể thiếu trong lĩnh vực VR và AR, nhờ khả năng theo dõi tay chính xác và tự nhiên. Các ứng dụng tiêu biểu gồm:

- **Varjo XR-3**: Một trong những kính VR cao cấp nhất, tích hợp Leap Motion Controller để hỗ trợ theo dõi bàn tay mà không cần sử dụng bộ điều khiển (controller). Điều này cho phép các nhà thiết kế công nghiệp và kỹ sư dễ dàng thao tác trực tiếp với các mô hình 3D.
- **STRATOS Inspire**: Một giải pháp kết hợp công nghệ Leap Motion với cảm giác xúc giác không chạm, được ứng dụng trong các trải nghiệm AR tương tác tại triển lãm hoặc quảng cáo.

Leap Motion Controller giúp tạo ra những bài học và chương trình đào tạo tương tác, cho phép học viên trải nghiệm thực tế hơn. **Flight Simulator Training** là một ứng dụng trong các chương trình mô phỏng huấn luyện phi công, Leap Motion Controller được sử dụng để kiểm soát các công tắc và thao tác trong buồng lái, giúp tăng cường kỹ năng mà không cần mô hình vật lý thực.

Hình 1.11. Ứng dụng Leap Motion trong mô phỏng lái máy bay

Và không thể không kể đến lĩnh vực giải trí, Leap Motion Controller đã mang lại trải nghiệm độc đáo cho người dùng khi được tích hợp trong một số sản phẩm nổi bật.**The Unspoken** là một tựa game VR nhập vai hành động, nơi người chơi hóa thân thành một pháp sư, sử dụng tay để thực hiện các phép thuật thông qua cử chỉ. Trò chơi này được phát triển bởi Insomniac Games và tích hợp hoàn hảo Leap Motion Controller, mang lại cảm giác chân thực khi thực hiện các thao tác [4].

Hình 1.12. Game The Unspoken sử dụng Leap Motion

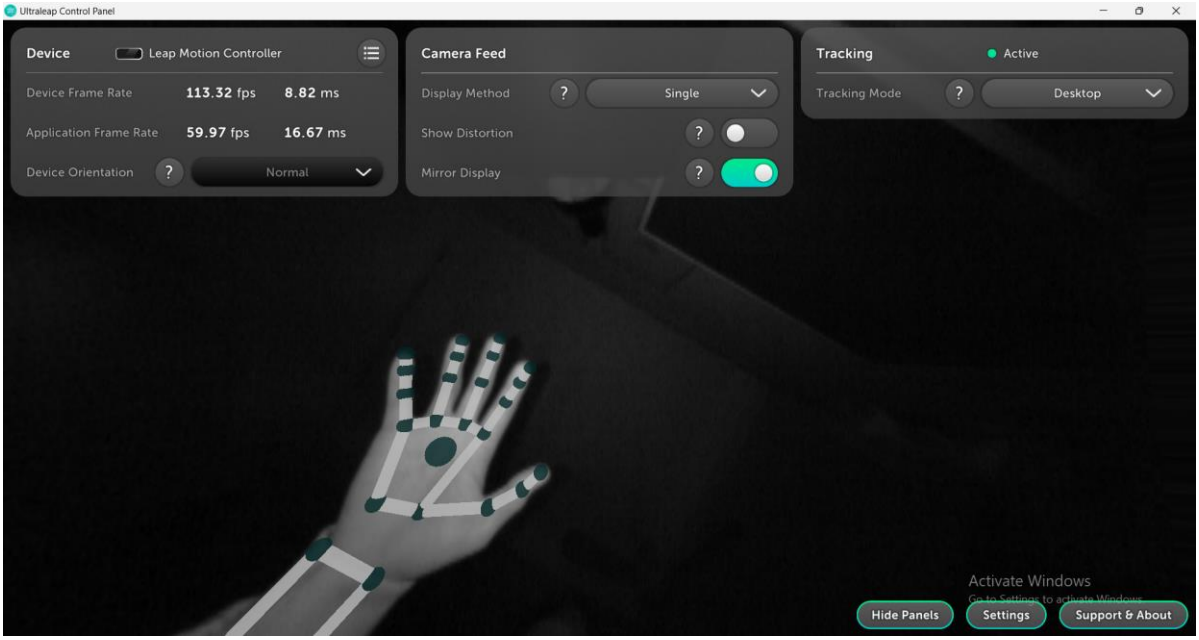
1.2.5. Cài đặt phần mềm

Ultraleap cung cấp phần mềm chính thức để hỗ trợ hiển thị, kiểm tra và theo dõi các thông số hoạt động khi Leap Motion Controller được kết nối với máy tính. Đây là một công

cụ quan trọng để kiểm tra trạng thái thiết bị và thực hiện các tùy chỉnh cần thiết cho trải nghiệm người dùng.

Hướng dẫn cài đặt và sử dụng:

- Truy cập đường dẫn chính thức và tải phần mềm phù hợp với hệ điều hành: https://leap2.ultraleap.com/downloads/leap-motion-controller/
- Cài đặt: Mở tệp cài đặt vừa tải về
- Kết nối: Sử dụng cáp USB đi kèm thiết bị để kết nối với máy tính, sau khi thiết bị được nhận có thể kiểm tra trạng thái trong **Ultraleap Control Panel**.



Hình 1.13. Giao diện Ultraleap Control Panel

Một số thông tin cơ bản trên phần mềm:

- **Device Frame Rate**: tốc độ khung hình thiết bị xử lý dữ liệu từ cảm biến
- **Application Frame Rate**: tốc độ khung hình của ứng dụng xử lý dữ liệu thu được từ Leap Motion.
- **Display Method**: Xác định cách hiển thị hình ảnh từ camera hồng ngoại. ***Side By Side*** là hiển thị hình ảnh từ từng camera riêng lẻ. ***Single*** là kết hợp hình ảnh từ 2 camera để có hình ảnh tổng thể.
- **Show Distortion**: Cho phép bật/tắt chế độ hiển thị hình ảnh bị bóp méo do ống kính của camera.
- **Mirror Display**: Bật/tắt chế độ phản chiếu hình ảnh.

- **Tracking Mode**: lựa chọn chế độ theo dõi phù hợp với tình huống sử dụng cụ thể của Leap Motion Controller. **Desktop Mode** khi thiết bị đặt nằm trên bàn phẳng, **Head Mounted** khi thiết bị được gắn trên kính thực tế ảo, **Screentop** khi được gắn trên cạnh của màn hình để theo dõi cử chỉ từ phía trước.

1.3. Kết luận chương

Chương 1 đã trình bày tổng quan về công nghệ nhận diện cử chỉ bàn tay và thiết bị Leap Motion Controller. Qua việc phân tích bài toán chung, nguyên lý hoạt động và khả năng ứng dụng của công nghệ nhận diện cử chỉ bàn tay, chúng ta nhận thấy đây là một lĩnh vực đầy tiềm năng, mở ra nhiều cơ hội trong các ngành như giải trí, y tế, giáo dục và nghiên cứu khoa học.

Leap Motion Controller, với khả năng theo dõi chuyển động tay chính xác và tốc độ cao, đã được giới thiệu như một trong những thiết bị tiên phong trong lĩnh vực này. Từ cấu tạo, nguyên lý hoạt động cho đến lịch sử phát triển, Leap Motion không chỉ là một thiết bị phần cứng mà còn mang lại những ứng dụng đa dạng, từ hỗ trợ học tập, thiết kế, đến chơi game và phát triển giao diện thực tế ảo (XR).

Những kiến thức này làm tiền đề quan trọng để bước sang chương tiếp theo, nơi sẽ đi sâu vào việc tích hợp Leap Motion với Unity. Qua đó, tập trung vào việc phát triển các ứng dụng thực tiễn, đặc biệt là trong lĩnh vực thiết kế và lập trình game dựa trên cử chỉ bàn tay.

CHƯƠNG 2. GIỚI THIỆU UNITY ENGINE VÀ ULTRALEAP PLUGIN

2.1. Giới thiệu về Unity Engine

2.1.1. Tổng quan

Unity được biết đến như là một công cụ trò chơi đa nền tảng, nó được phát triển bởi Unity Technologies. Mục đích sử dụng chủ yếu là để phát triển trò chơi điện tử và mô phỏng cho máy tính, thiết bị di động, bảng điều khiển,…Nhờ vào tính năng đa nền tảng, Unity là cái tên phổ biến với cả các nhà phát triển game tự do cũng như trong các studio game. Nó được dùng nhằm tạo ra những trò chơi như Hearthstone, Cuphead, Pokemon Go, Rimworld cùng vô vàn trò chơi khác nữa [5].

Hình 2.1. Logo Unity

Các sản phẩm của Unity 2D, 3D có thể được lập trình dựa trên 3 ngôn ngữ là C#, JavaScript và Boo. Trong đó, C# là ngôn ngữ chính mà rất nhiều lập trình viên Unity sử dụng cho tới thời điểm hiện tại. Unity đã có một lượng lớn người dùng cũng như sở hữu một thư viện tài nguyên khổng lồ. Không chỉ có tài liệu tuyệt vời, mà Unity còn có vô vàn video cùng những hướng dẫn trực tuyến đáng ngạc nhiên dành cho người dùng. Chính vì lý do này, Unity trở thành sự lựa chọn vô cùng sáng suốt cho người mới bước đầu tiếp cận với các công cụ game. Có mặt trong danh sách những công cụ trò chơi điện tử, Unity giữ vai trò tựa như một cổng thông tin tài nguyên và kiến thức, được xây dựng, phát triển chỉ dựa trên cộng đồng rộng lớn của riêng họ.

Như vậy, mục đích chính của Unity là tạo điều kiện thuận lợi cho các nhà phát triển để tạo ra trò chơi chất lượng cao, đa nền tảng, và mang tính thương mại. Unity giúp giảm thời gian và công sức phát triển, giúp tập trung vào việc sáng tạo và xây dựng trải nghiệm trò chơi độc đáo.

2.1.2. Ưu và nhược điểm của Unity

Ưu điểm của Unity:

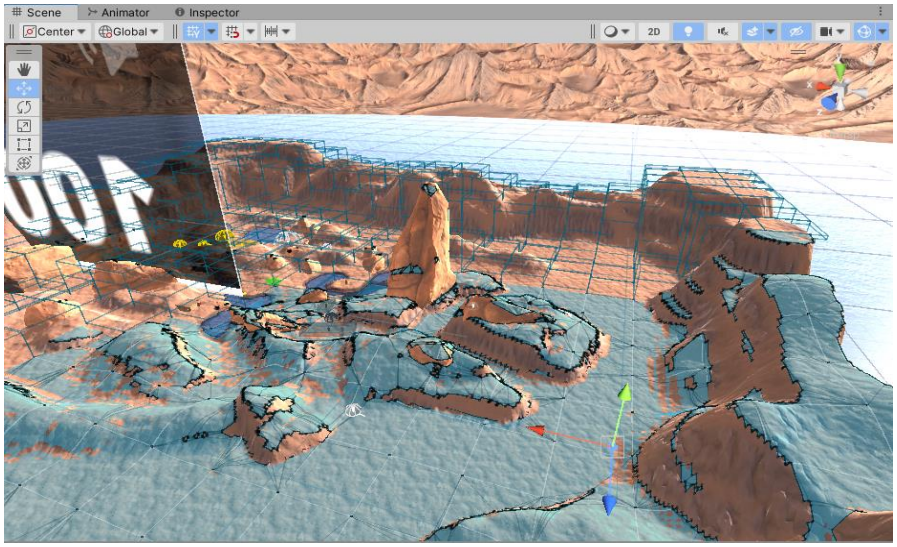
- Trên Editor, các nhà phát triển game không nhất thiết phải viết code nhằm sắp xếp những đối tượng trong game như các Engine khác. Thay vào đó, các Developer có thể kéo thả hoặc thay đổi vị trí của từng đối tượng trong game một cách trực tiếp.
- Có thể sử dụng đa nền tảng: Đây là một ưu điểm quan trọng vì nó giúp tiết kiệm nhiều công sức và chi phí cho doanh nghiệp vì game thể chạy được trên hầu hết hệ điều hành như Mobile (iOS, Android), Desktop (Window, Mac và Linux) hoặc Web (WebGL).
- Sử dụng miễn phí, đây là ưu điểm thu hút đông đảo Developer chọn làm việc với game engine này. Song, với việc game được tạo ra miễn phí thì điều bắt buộc là phải có Logo Unity trong game đó.
- Unity hỗ trợ rất nhiều định dạng asset khác nhau và có thể tự động di chuyển đến định dạng phù hợp nhất với nền tảng thích hợp.
- Khả năng dùng phổ biến C#
- Có công cụ rất trực quan, editor có thể mở rộng bằng plugins

Nhược điểm của Unity:

- Dung lượng Unity game bundle khá lớn: Khi đặt lên bàn cân với những game engine khác, game mà Unity sản xuất có dung lượng nặng và đây là điểm trừ lớn.
- Không thích hợp với những dự án lớn: không thể đào sâu quá vừa được xem là nhược điểm cũng vừa được xem là ưu điểm. Một mặt, unity cho phép quy trình nhanh chóng, thích ứng với những người mới bắt đầu, mặt khác điều đó được xem như unity không phải là thứ bạn tìm kiếm nếu bạn đang hy vọng sẽ tạo ra thứ gì đó thật đặc biệt, khác lạ hay trong một quy mô lớn.
- Mã nguồn của engine không được công bố dù cho những người dùng chấp nhận chi trả tiền. Điều đó có nghĩa là nếu bạn gặp một bug với engine thì lúc này bạn phải chờ unity fix chúng trong các bản tiếp theo. Từ đó có thể gây ra những vấn đề nghiêm trọng với project của bạn.

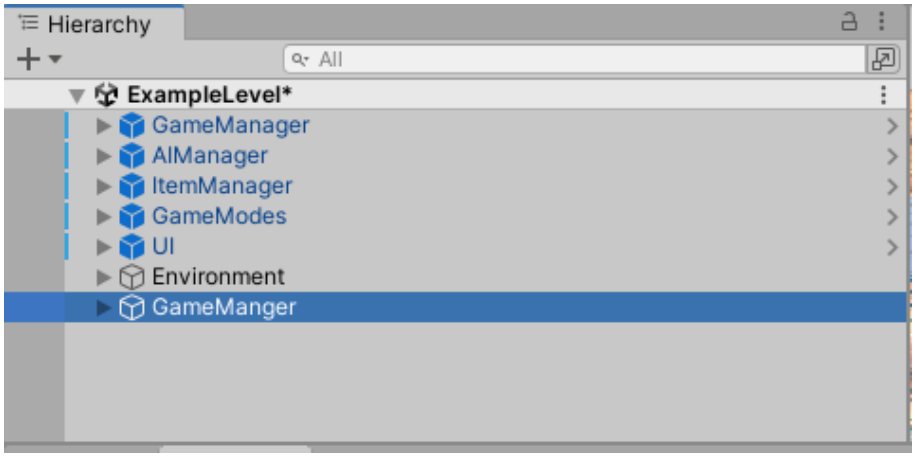
2.1.3. Một số thành phần chính trong Unity Editor

Cửa sổ Scene: Phần này phần hiển thị các đối tượng trong scene một cách trực quan, có thể lựa chọn các đối tượng, kéo thả, phóng to, thu nhỏ, xoay các đối tượng ... Phần này có để thiết lập một số thông số như hiển thị ánh sáng, âm anh, cách nhìn 2D hay 3D...



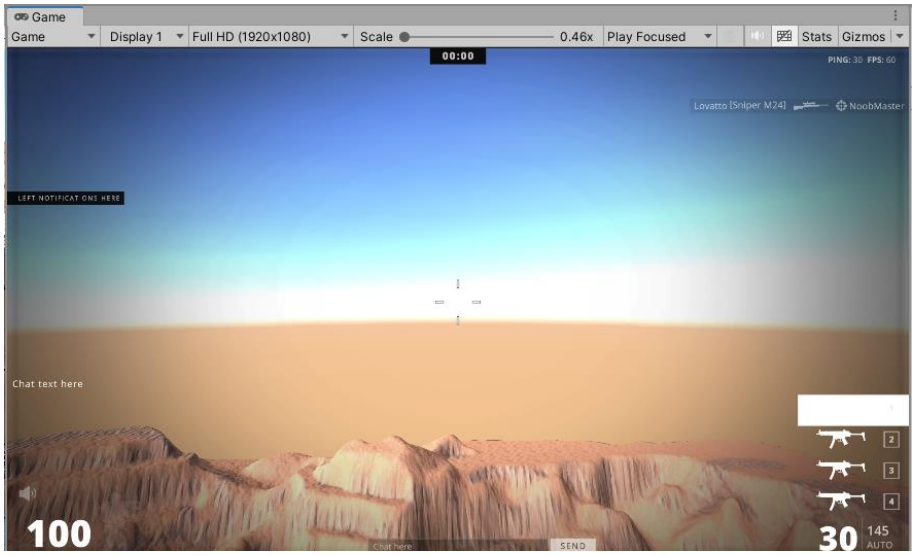
Hình 2.2. Cửa sổ Scene trong Unity

Cửa sổ Hierarchy: Tab hierarchy là nơi hiển thị các Gameobject trong Scenes hiện hành. Khi các đối tượng được thêm hoặc xóa trong Scenes, tương ứng với các đối tượng đó trong cửa sổ Hierarchy. Cửa sổ Hierarchy là một công cụ quan trọng giúp bạn quản lý và tương tác với tất cả các đối tượng trong Scene của mình trong Unity.



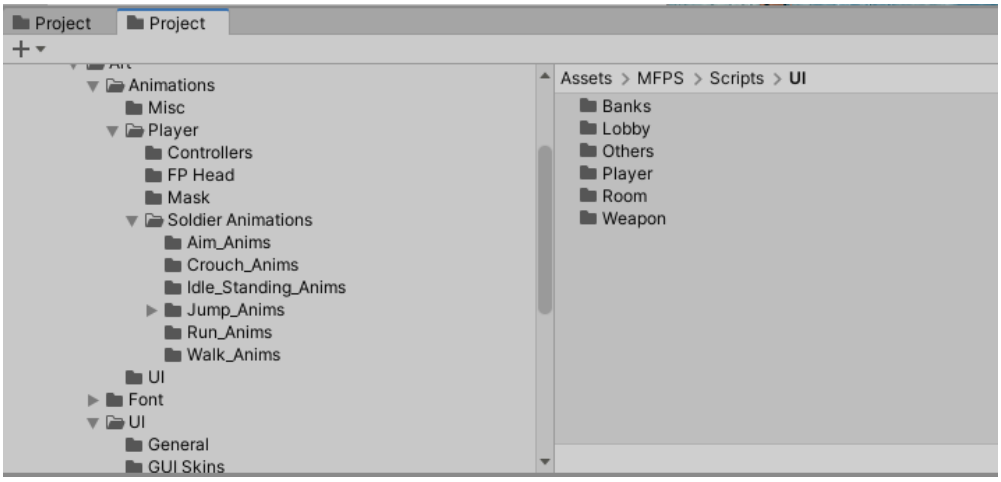
Hình 2.3. Cửa sổ Hierarchy trong Unity

Cửa sổ Game: Đây là màn hình demo Game, là góc nhìn từ camera trong game. Thanh công cụ trong cửa sổ game cung cấp các tùy chỉnh về độ phân giải màn hình, thông số (stats), gizmos, tùy chọn bật tắt các component…



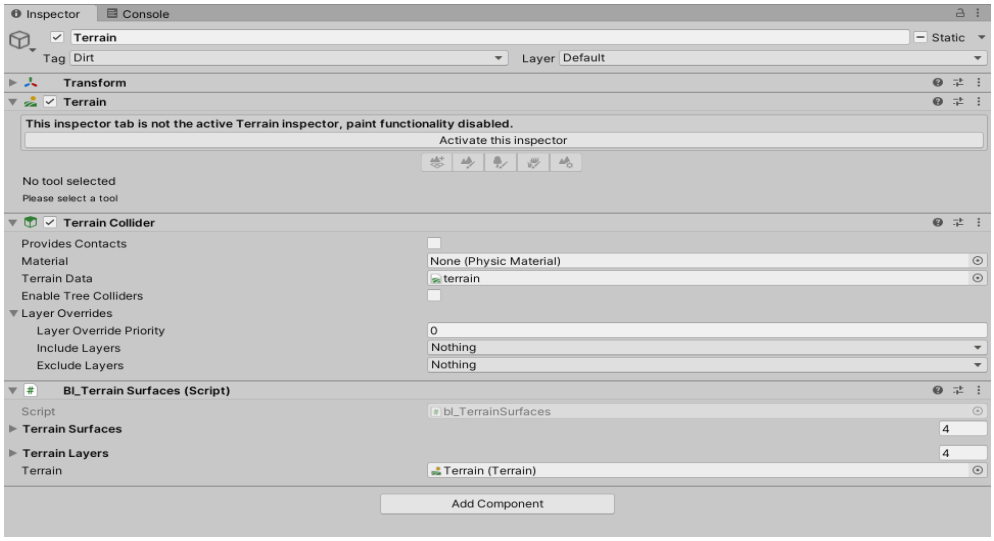
Hình 2.4. Cửa sổ Game trong Unity

Cửa sổ Project: Đây là cửa sổ explorer của Unity, hiển thị thông tin của tất cả các tài nguyên (Assets) trong game. Khi click vào một nhánh trên cây thư mục thì toàn bộ nội dung của nhánh đó sẽ được hiển thị ở khung bên phải.



Hình 2.5. Cửa sổ Project trong Unity

Cửa sổ Inspector: Cửa sổ Inspector hiển thị chi tiết các thông tin về Gameobject đang làm việc, kể cả những component được đính kèm và thuộc tính của nó. Bạn có thể điều chỉnh, thiết lập mọi thông số và chức năng của Gameobject thông qua cửa sổ Inspector.



Hình 2.6. Cửa sổ Inspector trong Unity

2.1.4. Một số khái niệm trong Unity Editor

GameObject: Một đối tượng cụ thể trong game gọi là một Gameobject, có thể là nhân vật, đồ vật nào đó. Ví dụ: cây cối, xe cộ, nhà cửa, người…

Component: Một GameObject sẽ có nhiều thành phần cấu tạo nên nó như là hình ảnh (sprite render), tập hợp các hành động (animator), thành phần xử lý va chạm (collision), tính toán vật lý (physical), mã điều khiển (script), các thành phần khác... mỗi thứ như vậy gọi là một component của GameObject.

Prefabs: Là một khái niệm trong Unity, dùng để sử dụng lại các đối tượng giống nhau có trong game mà chỉ cần khởi tạo lại các giá trị vị trí, tỉ lệ biến dạng và góc quay từ một đối tượng ban đầu. Ví dụ: Các đối tượng là đồng tiền trong game Mario đều có xử lý giống nhau, nên ta chỉ việc tạo ra một đối tượng ban đầu, các đồng tiền còn lại sẽ sử dụng prefabs. Hoặc khi ta lát gạch cho một cái nền nhà, các viên gạch cũng được sử dụng là prefabs.

Script: Script là tập tin chứa các đoạn mã nguồn, dùng để khởi tạo và xử lý các đối tượng trong game. Trong Unity có thể dùng C#, Javascript để lập trình Script.

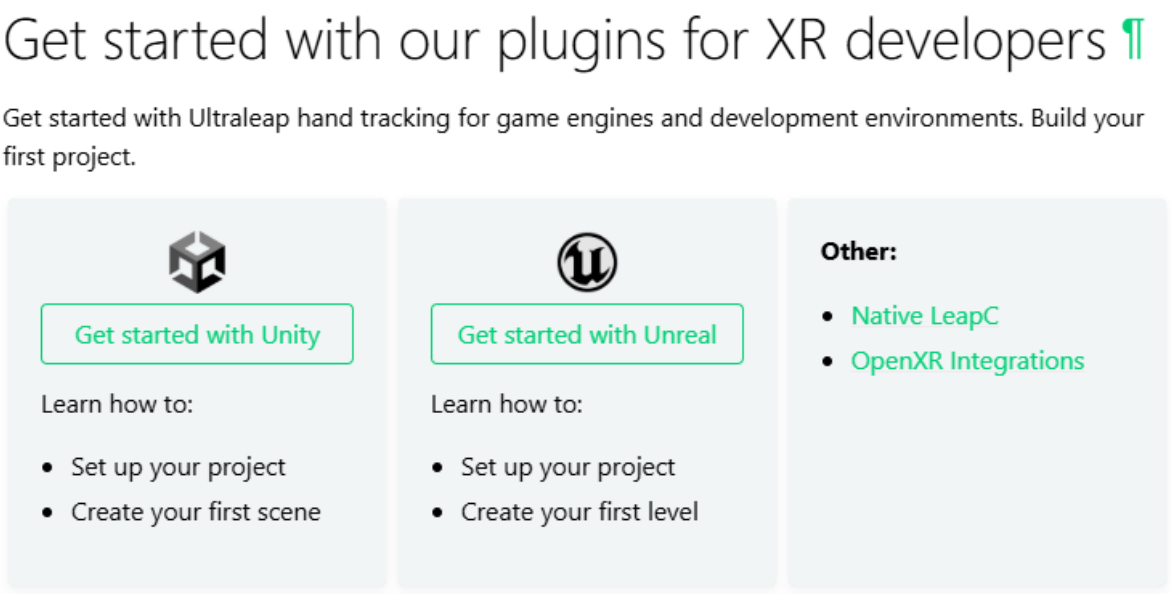
Scenes: Quản lý tất cả các đối tượng trong một màn chơi của game.

Assets: Bao gồm tất cả những gì phục vụ cho dự án game như sprite, animation, sound, script, scenes…

2.2. Ultraleap plugin trong Unity

2.2.1. Giới thiệu và cài đặt

Ultraleap cung cấp các plugin, công cụ (tools) và bộ tài liệu đầy đủ để hỗ trợ việc tích hợp thiết bị Leap Motion Controller vào các nền tảng phát triển XR (Extended Reality) và tabletop development. Ultraleap tương thích với các công cụ phát triển hàng đầu như Unity, Unreal Engine và nền tảng tiêu chuẩn OpenXR, cho phép nhà phát triển dễ dàng triển khai các trải nghiệm tương tác cử chỉ tay tự nhiên trong các dự án của mình [6].



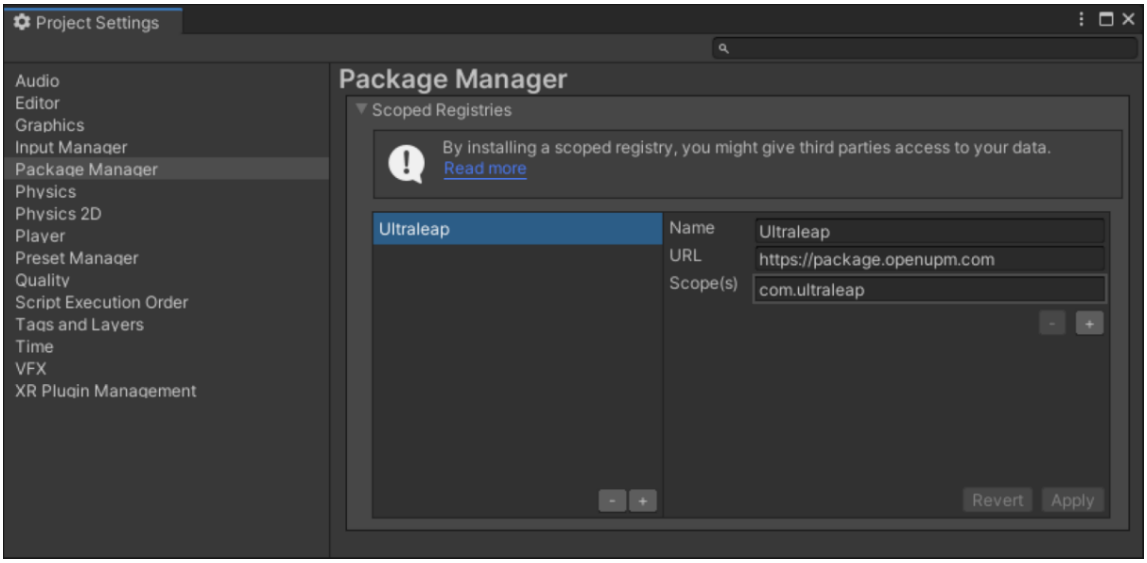
Hình 2.7. Giao diện trang chủ Ultraleap

Để tích hợp Ultraleap vào Unity cần đảm bảo các điều kiện sau:

- Thiết bị nhận diện cử chỉ tay của Ultraleap (Leap Motion Controller)
- Máy tính có thể kết nối được với thiết bị qua cổng USB
- Phần mềm Ultraleap Control Panel
- Unity Editor 2021 LTS hoặc mới hơn

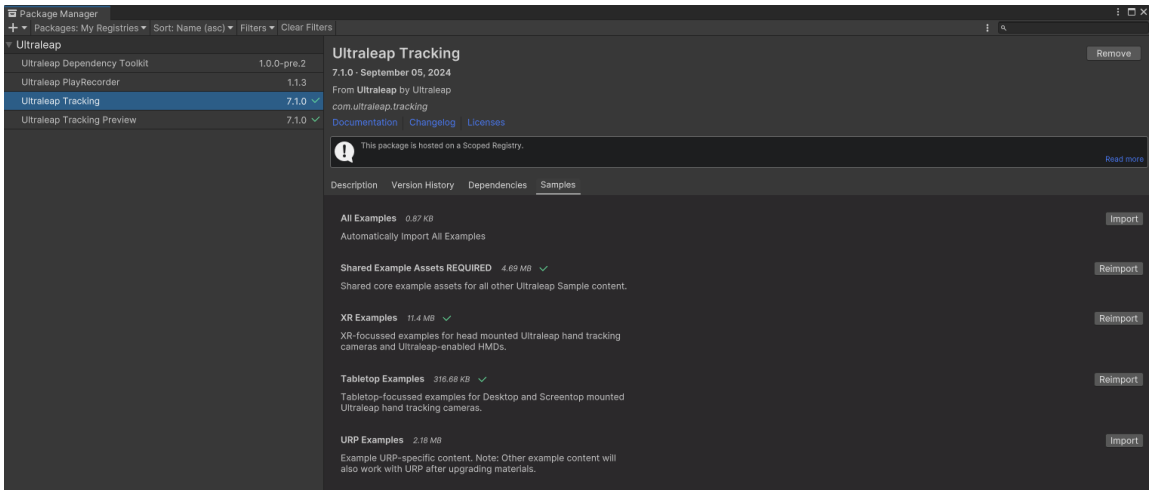
Tích hợp plugin Ultraleap vào Unity:

1. Xoá mọi module Ultraleap hiện có trong dự án
2. Trong Unity, chọn Edit/Project Settings/Package Manager, thêm registry của Ultraleap



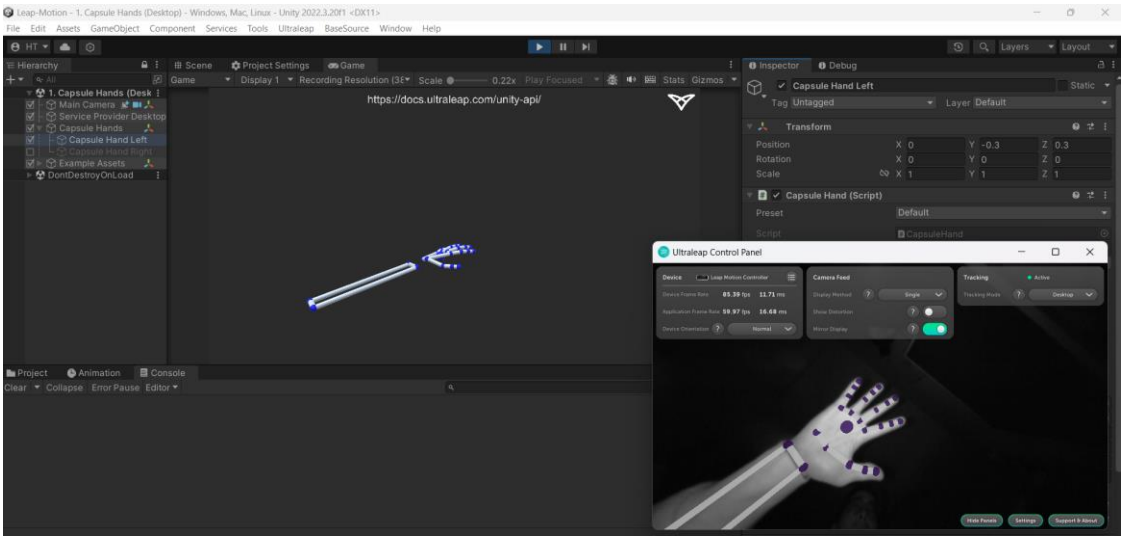
Hình 2.8. Thêm Registry của Ultraleap trong Unity

3. Mở cửa sổ Package Manager, ở mục My Registries, các plugin kèm các ví dụ của Ultraleap đã sẵn sàng để tải về và import vào dự án.



Hình 2.9. Các package trong Ultraleap plugin

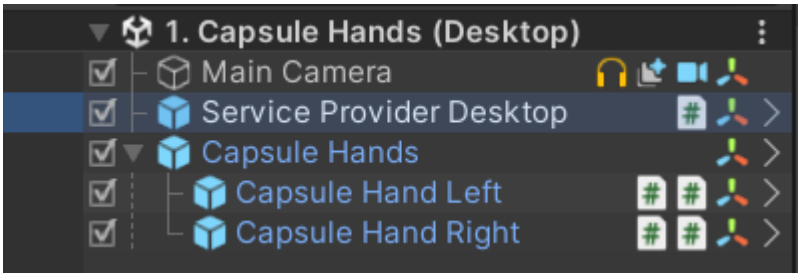
4. Kết nối Leap Motion Controller với máy tính và chạy thử các example trong folder **Samples/Ultraleap Tracking/7.0.1/Tabletop Examples**. Các example cung cấp cái nhìn ban đầu, cách setup và các thành phần trên scene cùng với các tính năng cơ bản của Ultraleap.



Hình 2.10. Chạy scene demo trong Ultraleap plugin

2.2.2. Các thành phần cơ bản

Khi tích hợp Leap Motion Controller với Unity, có 2 thành phần cơ bản cần được thiết lập trên scene để hoạt động chính xác là **Service Provider** và **Hands**.



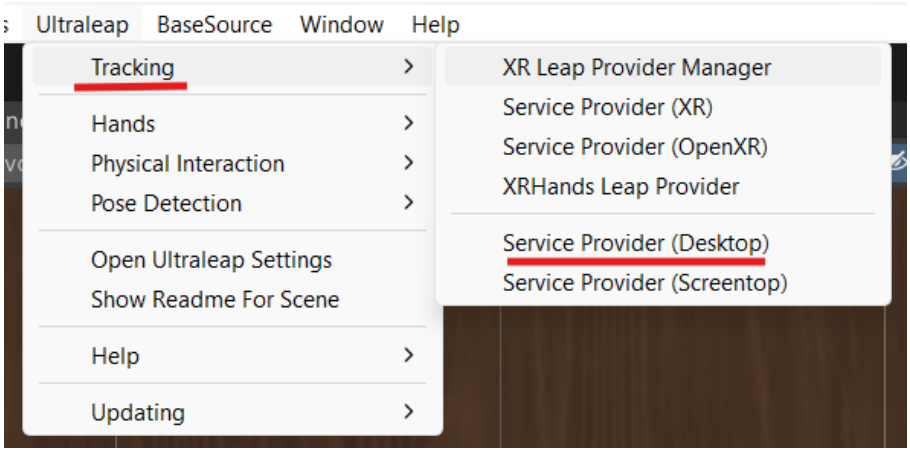
Hình 2.11. Thành phần cơ bản cần có

Service Provider là thành phần cốt lõi trong Ultraleap Unity Plugin, đóng vai trò trung gian giữa thiết bị Leap Motion Controller và các model tay trong Unity. Thành phần này đảm bảo việc truyền dữ liệu từ thiết bị sang Unity và cập nhật theo thời gian thực.

Vai trò chính của Service Provider là:

- Nhận dữ liệu từ tay: Service Provider nhận dữ liệu cử chỉ bàn tay và ngón tay từ Leap Motion Controller thông qua phần mềm.
- Xử Lý Khung Hình (Frames): Cung cấp khung hình (frame) hiện tại bao gồm thông tin vị trí, góc xoay, hướng và trạng thái của các bàn tay được nhận diện.
- Cập Nhật Dữ Liệu Real-Time: Truyền dữ liệu cử chỉ tay tới các Hand Models để hiển thị và tương tác trong Scene Unity theo thời gian thực.

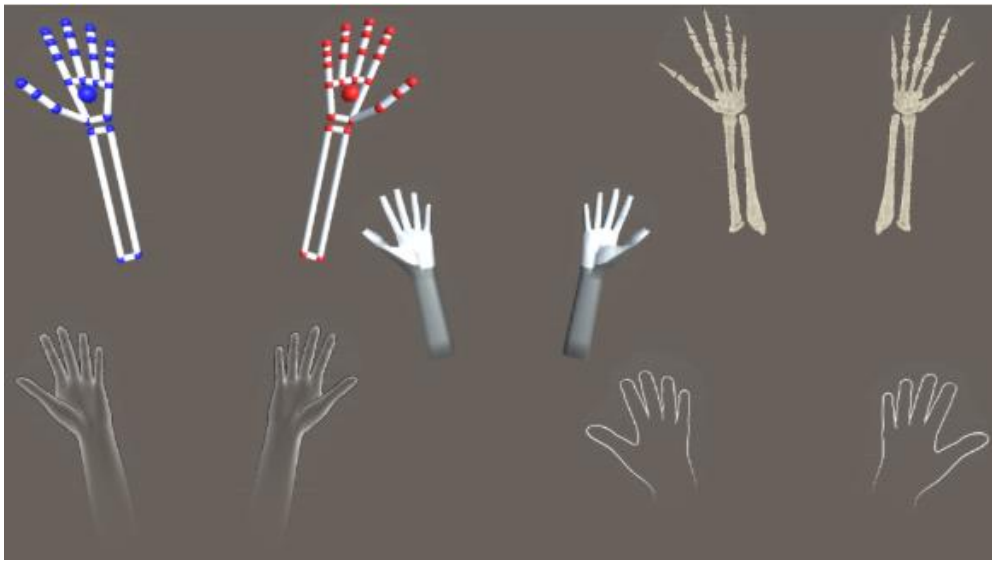
Ultraleap cung cấp nhiều Service Provider tuỳ tình huống sử dụng thiết bị như Desktop mode, Screentop hay XR (dành cho môi trường thực tế ảo).



Hình 2.12. Thêm Service Provider qua MenuItem

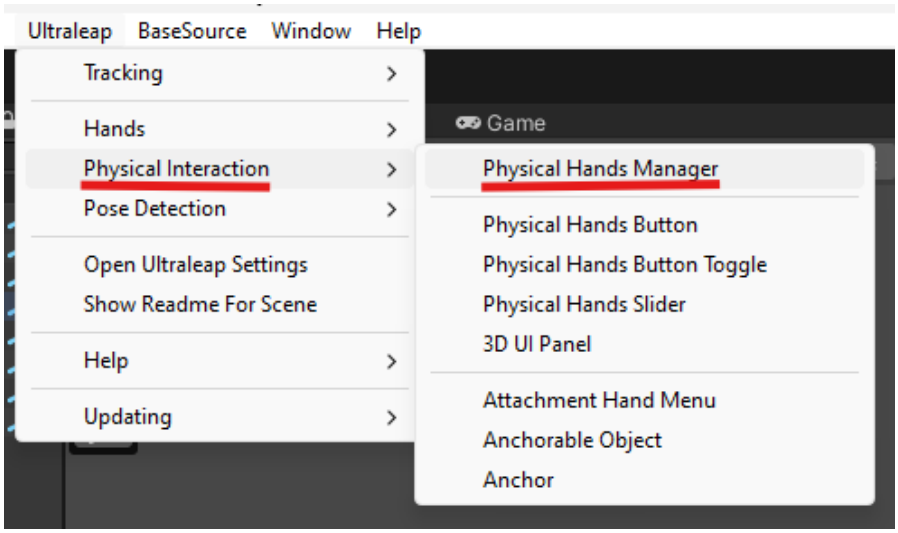
Cùng với Service Provider thì **Hand Models** đại diện cho mô hình bàn tay ảo trong Unity, hiển thị và cập nhật cử chỉ tay theo thời gian thực dựa trên dữ liệu nhận được từ Service Provider.

Ultraleap cung cấp sẵn một số mô hình tay, và có thể thêm lên trên scene bằng menu item *Ultraleap/Hands/...*



Hình 2.13. Các Hands mà Ultraleap cung cấp

Physical Hands Manager là một component trong Ultraleap Unity Plugin dùng để tạo và quản lý các bàn tay vật lý (Physical Hands) trong Unity. Thành phần này đặc biệt hữu ích khi bạn muốn kết hợp Leap Motion Controller với Rigidbody và Collider để tạo ra các bàn tay có thể tương tác vật lý với các đối tượng trong môi trường.

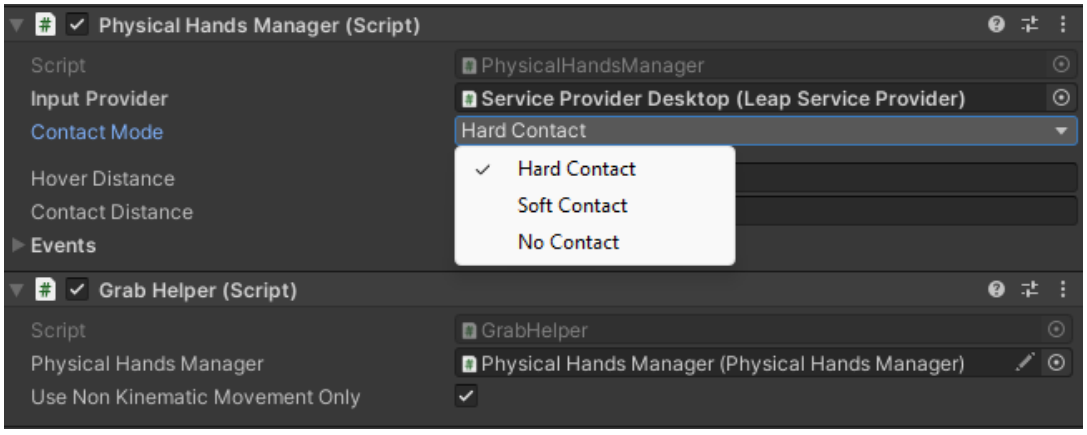


Hình 2.14. Thêm Hands lên scene bằng Menu Item

Physical Hand sử dụng engine vật lý tích hợp của Unity, do đó có thể tương tác được với bất kỳ đối tượng nào có Rigibody và Collider. Physical Hand Manager tự động kết nối với Leap Service Provider để nhận dữ liệu cử chỉ tay theo thời gian thực và áp dụng lên các bàn tay vật lý.

Physical Hands Manager có 3 chế độ kết nối:

- **Hard Contact**: Tay sẽ bao quanh các vật thể thay vì đi xuyên qua. Như này tay có thể cầm hoặc đẩy được vật.
- **Soft Contact**: Tay sẽ xuyên qua các đồ vật nhưng vẫn cầm nắm được như bình thường.
- **No Contact**: Tay sẽ xuyên qua và không tương tác gì với các vật thể.

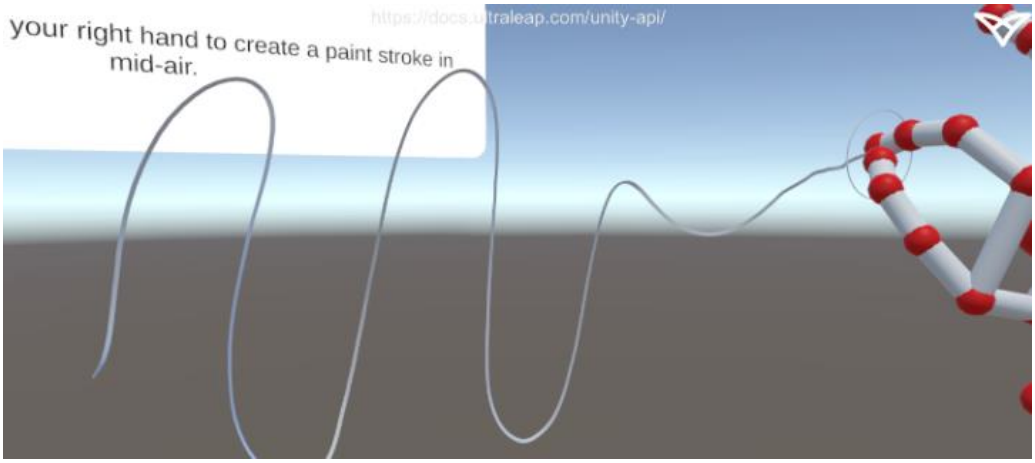


Hình 2.15. Các chế độ tương tác vật lý

2.2.3. Các tính năng khác

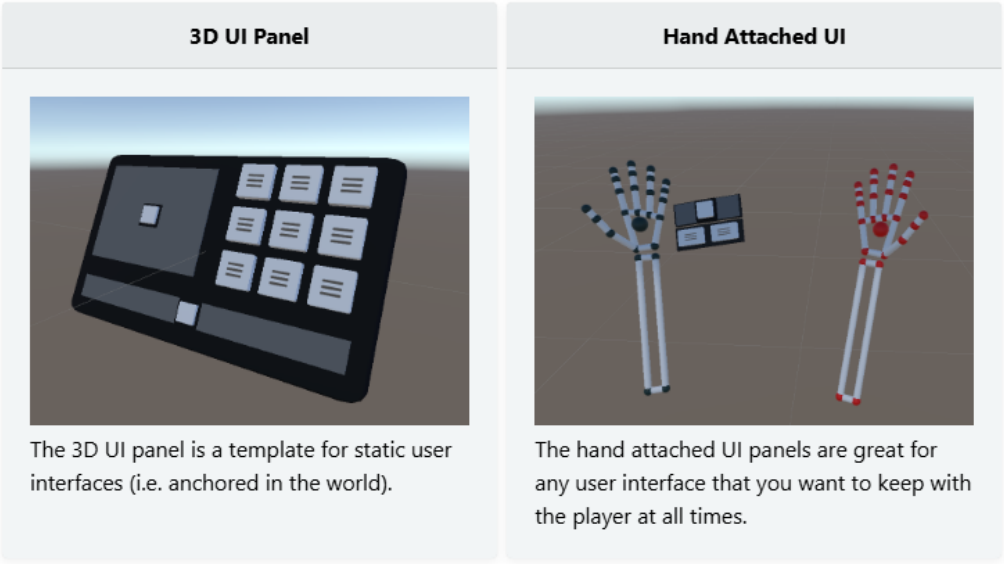
Ngoài những thành phần cơ bản trên, Ultraleap Unity Plugin còn cung cấp nhiều chức năng quan trọng khác để mở rộng khả năng tương tác và phát triển ứng dụng:

- Kiểm tra xem có đang tóm lấy vật (Pinch) hoặc đang kéo theo vật (Grab).
- Kiểm tra dáng của tay dựa trên một số dáng tay đã có sẵn, và hoàn toàn có thể tạo thêm các dáng tay mới.
- Chụm ngón tay để vẽ (Pinch to Paint)



Hình 2.16. Pinch Paint trong Ultraleap

- Tương tác với các UI 3D như là các nút (button), thanh kéo (slider), công tắc (toggle) hoặc các bảng điều khiển (panel).



Hình 2.17. Tương tác UI 3D trong Ultraleap

2.3. Kết luận chương

Chương 2 đã cung cấp cái nhìn toàn diện về Unity Engine và Ultraleap Plugin, hai thành phần cốt lõi trong việc xây dựng trò chơi tích hợp Leap Motion Controller.

Chương này đã làm rõ các khái niệm tổng quan, mục đích phát triển, cùng những ưu và nhược điểm của Unity Engine. Đồng thời, nội dung cũng giới thiệu chi tiết các thành phần chính và một số khái niệm cơ bản trong Unity Editor, tạo nền tảng vững chắc để phát triển game.

Về Ultraleap Plugin, chương này đã trình bày từ cách cài đặt, các thành phần cơ bản như Service Provider và Physical Hands Manager, cho đến các tính năng nâng cao khác. Các tính năng này không chỉ hỗ trợ việc tích hợp Leap Motion vào Unity mà còn mở rộng tiềm năng sáng tạo trong thiết kế và phát triển game dựa trên cử chỉ bàn tay.

CHƯƠNG 3. THIẾT KẾ GAME CHÉM HOA QUẢ

3.1. Hình thành ý tưởng game

Phân tích các khả năng của thiết bị Leap Motion Controller có thể làm được khi tích hợp vào Unity để thu hẹp thể loại, lối chơi, đặc điểm của trò chơi để chọn ra trò chơi sẽ phát triển trong đồ án.

Leap Motion Controller hướng đến phát triển các sản phẩm có sự tương tác trực tiếp từ bàn tay vì có thể theo dõi chính xác vị trí, tốc độ, và hướng di chuyển của từng ngón tay hoặc cả bàn tay trong không gian 3D. Khi được tích hợp và sử dụng ở chế độ để bàn (tabletop): đặt thiết bị trên mặt bàn và thao tác tay bên trên thiết bị, tầm theo dõi của thiết bị với tay sẽ bị giới hạn trong độ cao khoảng 60cm và góc khoảng 140 độ.

Dựa trên các yếu tố trên, trò chơi hướng tới phát triển sẽ có các đặc điểm sau:

- Thể loại game phù hợp:
  - Game phản xạ nhanh (Action Arcade).
  - Game giải đố tương tác (Puzzle).
  - Game mô phỏng điều khiển tay (Simulation).
  - Game chiến thuật theo lượt với thao tác kéo thả.
- Đặc điểm lối chơi:
  - Tương tác trực tiếp bằng cử chỉ tay: kéo, vuốt, hoặc chạm.
  - Tận dụng khả năng đa ngón tay và theo dõi 3D để tạo sự chân thực.
  - Phạm vi chơi hẹp, yêu cầu thiết kế vật thể trong tầm với của người chơi.
- Ví dụ về dạng trò chơi:
  - Game phản xạ: Chém hoa quả (Fruit Ninja).
  - Game xếp hình: Tetris 3D hoặc xếp khối theo bàn tay.
  - Game chiến thuật: Di chuyển quân cờ trên bàn.
  - Game mô phỏng: Đàn piano ảo, nấu ăn bằng tay.

Chém hoa quả (Fruit Ninja [7]) đáp ứng được đầy đủ các yếu tố trên để có thể trở thành một trò chơi tích hợp Leap Motion Controller. Được phát hành năm 2010 trên các nền tảng Android và iOS, tính đến nay đã được phát triển trên nhiều nền tảng khác như Xbox 360, Windows, … trò chơi đã chứng minh được sự thành công của mình với độ phổ biến cao và dễ tiếp cận phù hợp với mọi lứa tuổi, hiệu ứng âm thanh bắt mắt thu hút người chơi, cùng nhiều chế độ chơi đa dạng và cuốn hút.

Chém hoa quả yêu cầu thao tác chém nhanh và chính xác, phù hợp với khả năng nhận diện cử chỉ và tốc độ của Leap Motion. Động tác chém, vuốt trong game tương ứng với hành động tay thực tế, mang lại trải nghiệm nhập vai. Trái cây xuất hiện trong không gian giới hạn, dễ điều chỉnh để nằm trong tầm nhận diện của Leap Motion (60cm, 140 độ), đồng thời tay cũng không yêu cầu di chuyển ra ngoài vùng hoạt động, tránh được hạn chế kỹ thuật của Leap Motion.

3.2. Thiết kế các đối tượng trong game

Quả (Fruit):

- Có dạng hình tròn hoặc bầu dục.
- Có vật lý, xoay tròn, được bắn lên.
- Các quả khác nhau có kích thước, điểm số khác nhau.
- Khi cắt vào, quả chia đôi thành 2 nửa có kích thước đúng với góc cắt.
- Khi cắt và chia quả thành 2 nửa kích thước gần bằng nhau, tính là cắt hoàn hảo (perfect) và nhân đôi điểm số.
- Quả tồn tại trong 1 khoảng thời gian nhất định, sau khi cắt xong hoặc đã bay ra khỏi màn hình 1 khoảng thời gian thì bị xoá (destroy).

Bom (Bomb):

- Có vật lý tương tự như quả.
- Khi cắt vào bomb gây nổ và trừ mạng, khi hết mạng sẽ thua.
- Có tỉ lệ nhất định sinh ra bomb, vừa đủ để điều chỉnh được độ khó game.
- Cũng tồn tại trong 1 khoảng thời gian tương tự quả.

Hệ thống sinh quả (FruitSpawner):

- Có thời gian delay giữa các lần sinh quả
- Quả được sinh ra trong 1 vùng.
- Có lực bắn các quả lên khi quả được sinh ra.
- Quả được bắn lên theo góc ngẫu nhiên, bay lên và rơi xuống bởi trọng lực.

Dao chém (Blade):

- Chém được quả và bom.
- Vị trí của đường chém được cập nhật theo vị trí đầu ngón tay trỏ của bàn tay.
- Có hiệu ứng trail.

Quản lý điểm (ScoreManager):

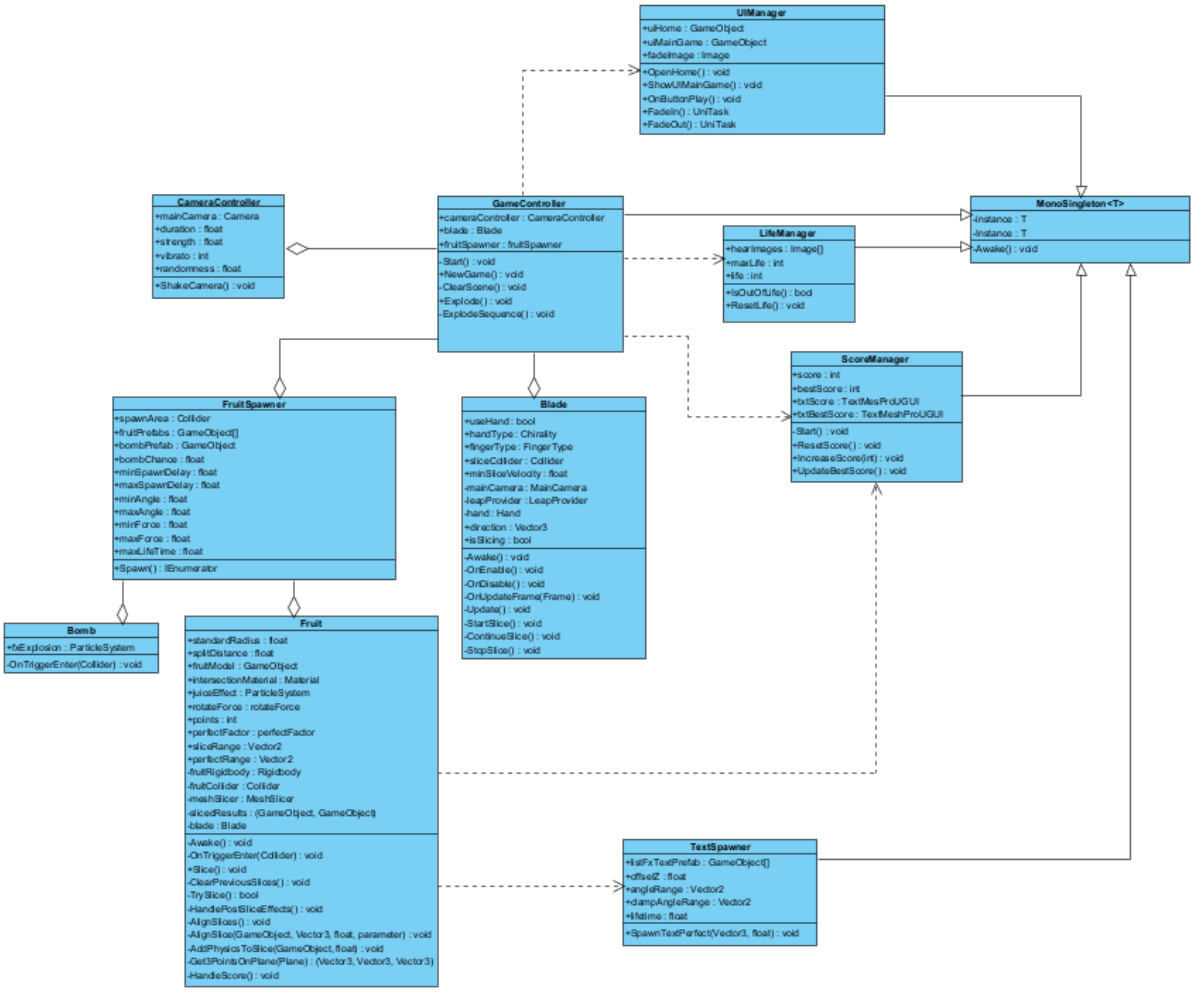
- Quản lý và hiển thị điểm trong màn chơi.

- Reset điểm khi bắt đầu một màn mới.
- Tăng điểm khi cắt được quả.
- Cập nhật và lưu điểm số cao nhất đạt được.

Quản lý mạng (LifeManager):

- Quản lý và hiển thị mạng trong màn chơi, tối đa 3 mạng.
- Giảm mạng khi chém phải bomb.
- Kết thúc game khi không còn mạng.

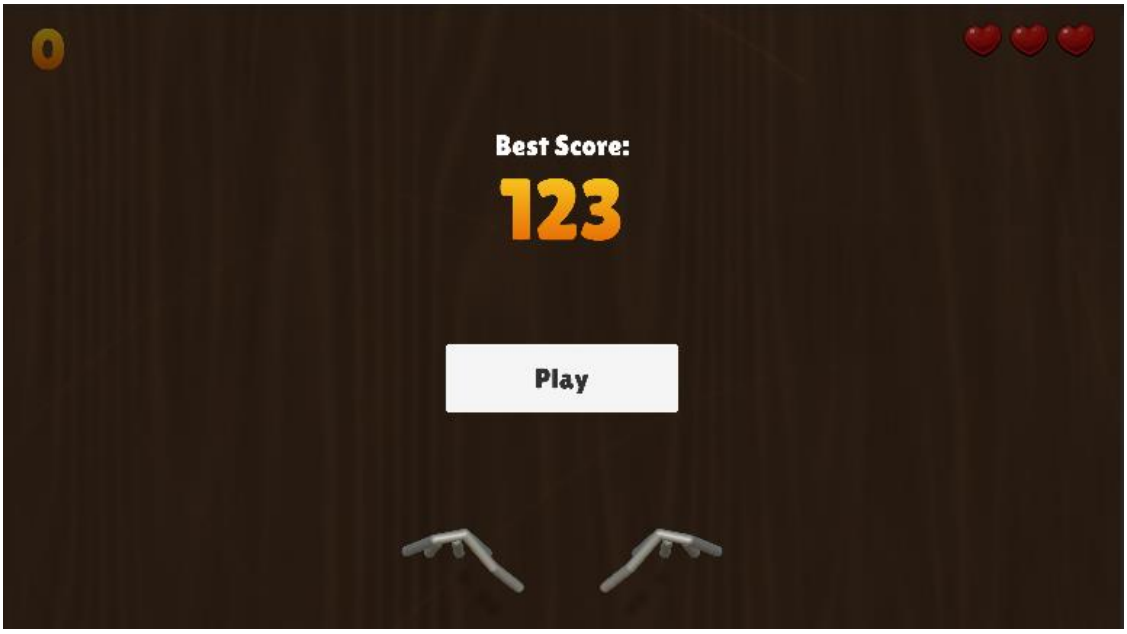
3.3. Biểu đồ lớp



Hình 3.1. Biểu đồ lớp

3.4. Thiết kế giao diện

Giao diện Home trước khi vào màn chơi:



Hình 3.2. Giao diện Home

Giao diện trong lúc chơi:

Hình 3.3. Giao diện GamePlay

3.5. Tổng kết chương

Chương 3 đã trình bày quá trình thiết kế trò chơi chém hoa quả tích hợp Leap Motion, tập trung vào các bước từ hình thành ý tưởng đến chi tiết thiết kế.

Phần đầu chương làm rõ ý tưởng phát triển game dựa trên khả năng nhận diện cử chỉ bàn tay của Leap Motion. Với mục tiêu tạo ra một trò chơi đơn giản nhưng hấp dẫn, phù hợp với điều kiện phần cứng và phạm vi dự án, trò chơi được định hình xoay quanh thao tác chém hoa quả trực quan.

Tiếp đó, chương này đi sâu vào thiết kế các đối tượng trong game, như quả, bom và các yếu tố giao diện. Luồng hoạt động được thiết kế rõ ràng, đảm bảo logic và trải nghiệm liền mạch. Biểu đồ lớp được xây dựng để mô tả cấu trúc phần mềm, giúp định hình cách các thành phần trong game tương tác với nhau.

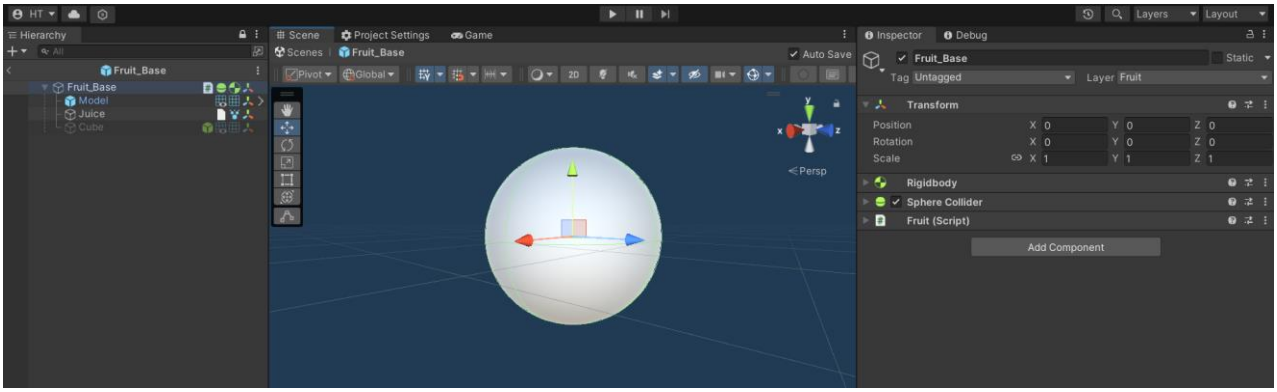
Cuối cùng, phần thiết kế giao diện tập trung vào việc xây dựng các màn hình trực quan, thân thiện với người dùng, đảm bảo hỗ trợ tốt nhất cho trải nghiệm chơi game.

Những nội dung trong chương 3 đã đặt nền móng vững chắc cho việc xây dựng game ở chương tiếp theo, nơi sẽ tập trung vào việc hiện thực hóa các thiết kế này thành sản phẩm hoàn chỉnh.

CHƯƠNG 4. XÂY DỰNG GAME UNITY TÍCH HỢP LEAP MOTION

4.1.Xây dựng hệ thống tạo quả và bom

4.1.1. Tạo prefab quả



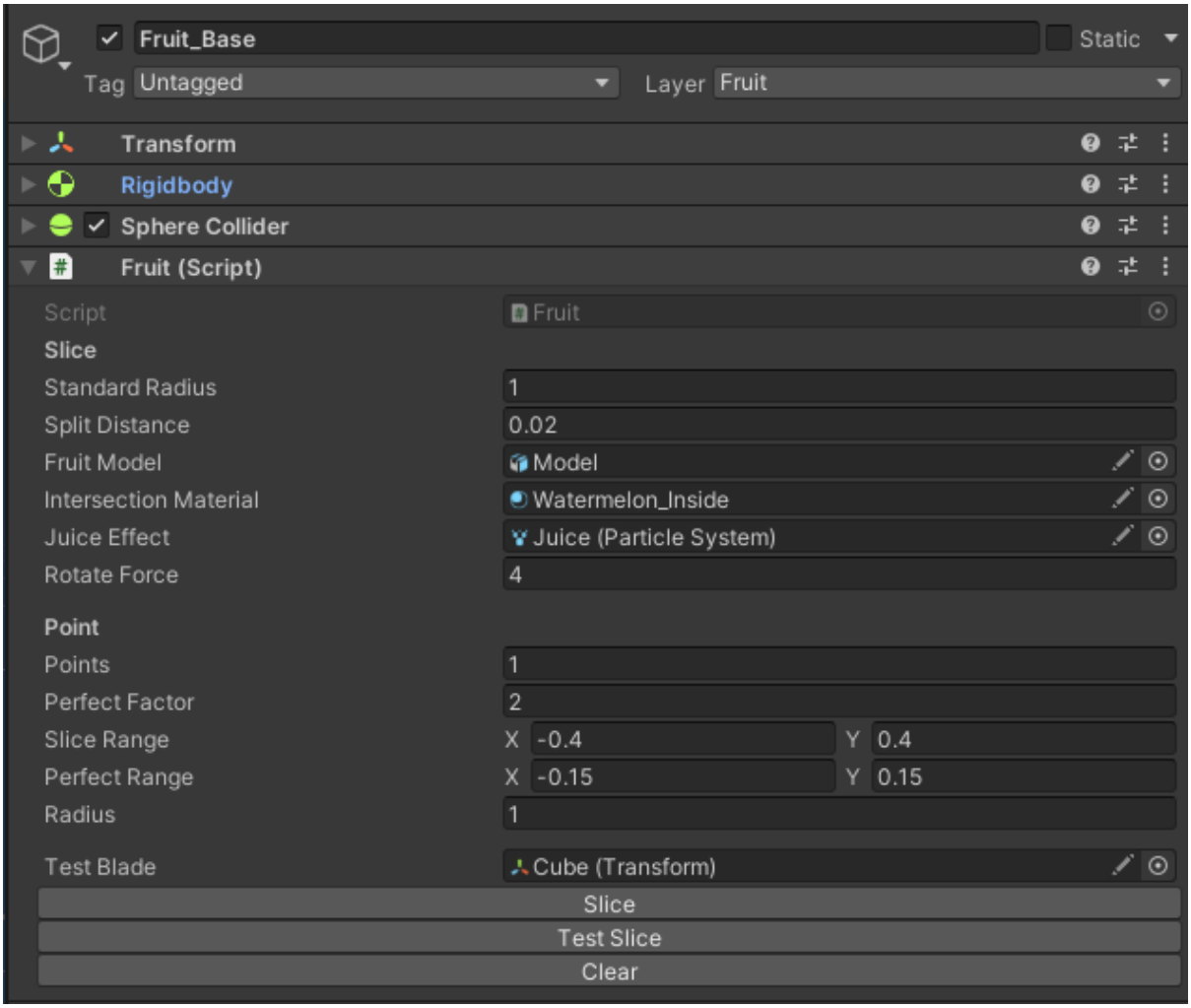
Hình 4.1. Prefab fruit gốc

Cấu trúc của 1 đối tượng Fruit bao gồm:

- Model 3D: Tạo một Sphere (khối cầu) là con của đối tượng. Hiện tại Sphere này sẽ đại diện cho model quả luôn, đến quá trình hoàn thiện sản phẩm và có model các loại quả cụ thể từ 3D artist ta sẽ thay vào sau.
- Tạo các material làm màu sắc ngoài của các quả cũng như màu sắc cho lát cắt bên trong khi quả bị chém đôi.

Hình 4.2. Các material cho quả

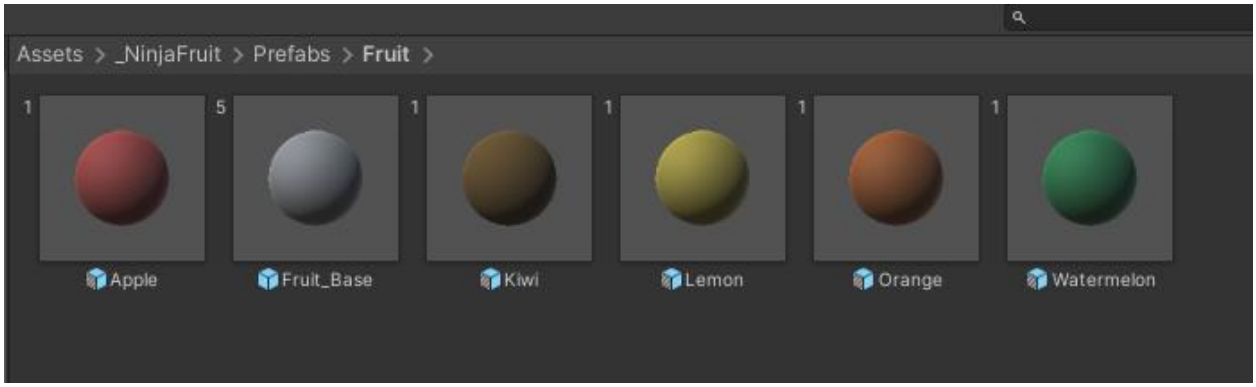
- Juice: sử dụng Particle System để tạo ra 1 hiệu ứng toé nước khi quả bị cắt. Sử dụng màu của hiệu ứng tương ứng theo màu bên ngoài của quả luôn.
- Sau khi đã tạo xong đối tượng Fruit, ta thêm các component Rigidbody, Sphere Collider để tạo vật lý, tạo và thêm script Fruit để xử lý các logic.



Hình 4.3. Các component trong prefab quả

Đối tượng Fruit_Base sẽ để là prefab quả gốc và tạo ra các quả khác là prefab variant từ Fruit_Base. Điều này giúp dễ dàng sửa đổi hoặc thay đổi thuộc tính chung giữa các quả nhờ việc chỉ chỉnh sửa Fruit_Base và các quả khác cũng sẽ được cập nhật theo, lại vẫn đảm bảo được có thể thay đổi màu sắc, thuộc tính riêng ở từng quả cụ thể.

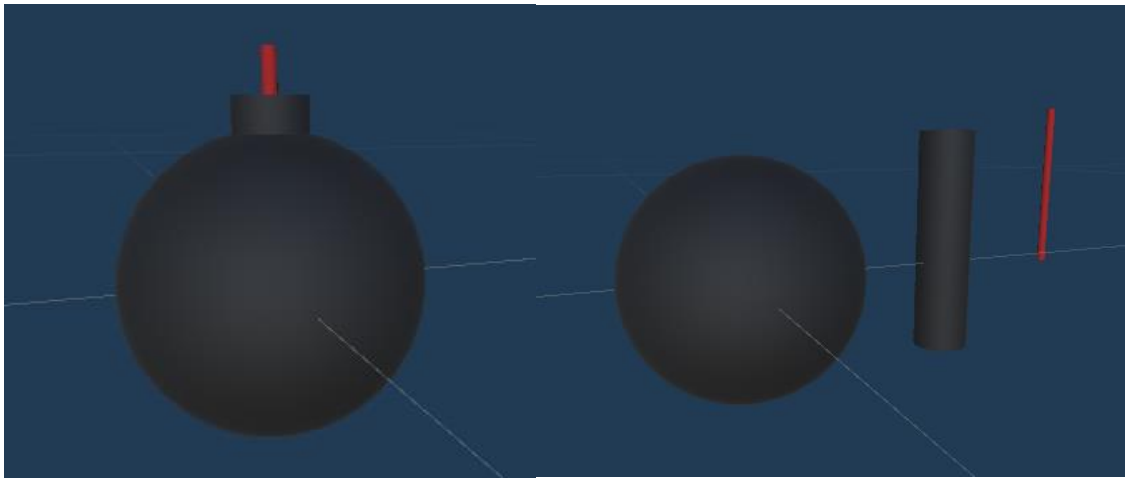
Tạo 5 loại quả với màu sắc, kích thước từ nhỏ đến to để tạo mức độ khó - dễ trong quá trình chơi.



Hình 4.4. Các prefab quả trong game

4.1.2. Tạo prefab bomb và logic

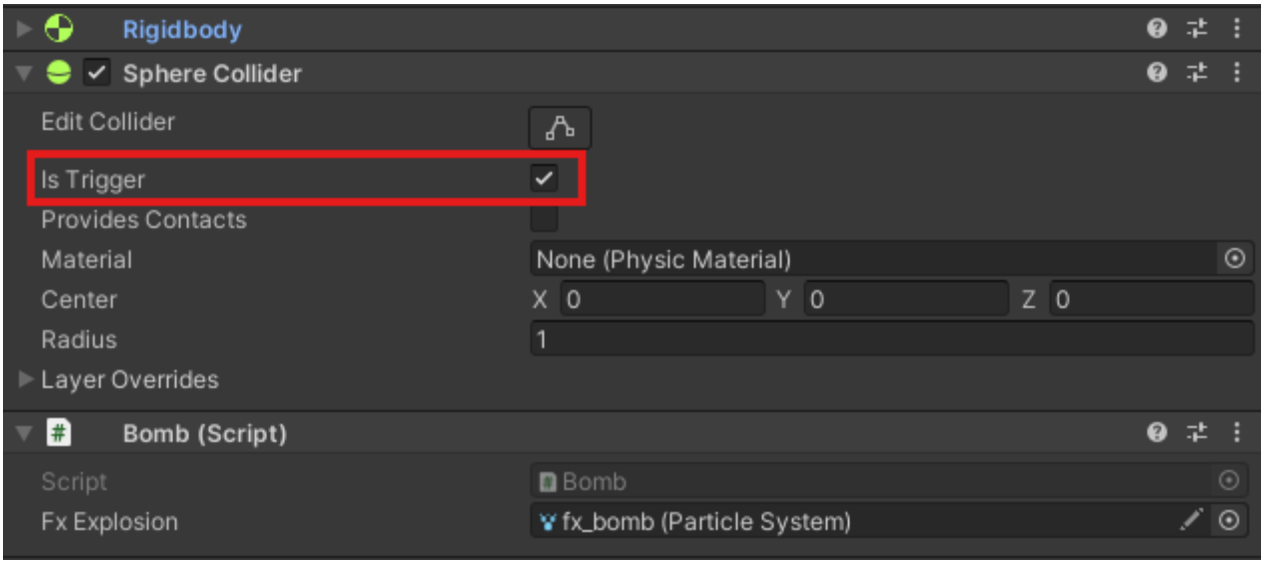
Tạo model cho quả bomb với 3 thành phần cơ bản: 1 Sphere (khối cầu) và 2 Cylinder (khối trụ)



Hình 4.5. Cấu tạo model bomb

Tương tự như quả, bomb cũng được thêm các component Rigidbody và Sphere Collider để có thể tương tác vật lý. Kèm theo là một Particle System tạo hiệu ứng khói và lửa khi bomb nổ.

Khi có Collider, ta có thể bắt được các sự kiện va chạm của đối tượng. Mục tiêu là đường cắt cũng sử dụng Collider, như vậy ta có thể kiểm tra được sự kiện đường cắt chạm vào bomb cũng như các quả. Vì chỉ muốn xử lý các logic nổ, trừ mạng, … tại thời điểm cắt vào bomb, mà không muốn bomb làm thay đổi quỹ đạo của đường cắt như 2 vật thể cứng va chạm, ta đặt bomb có Collider là trigger (va chạm xuyên qua nhau).



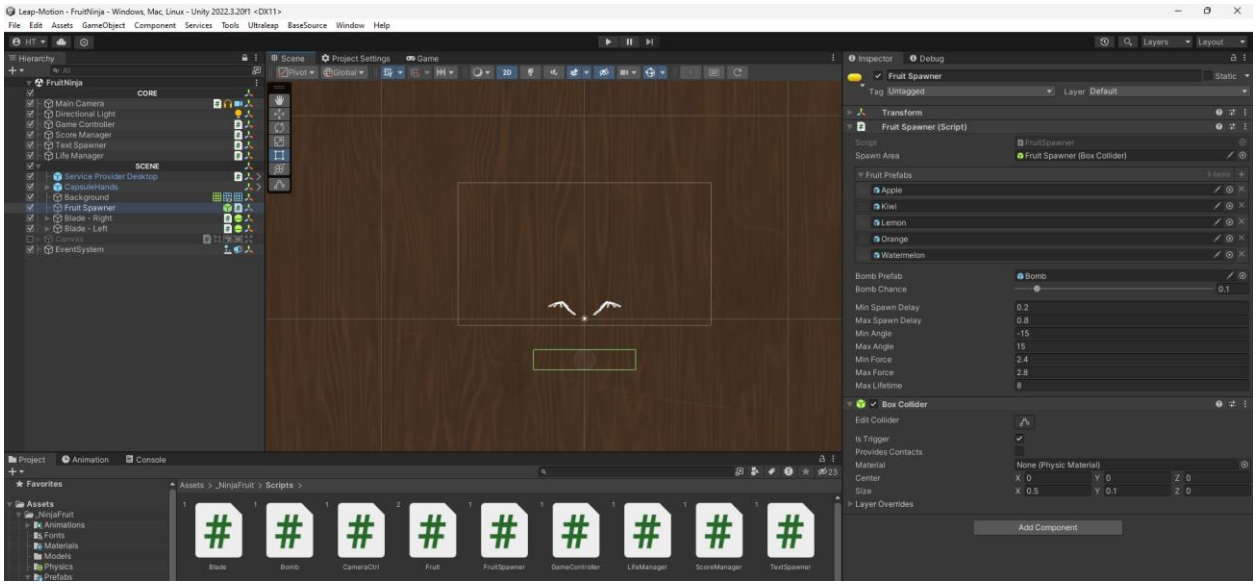
Hình 4.6 Collider trigger trong bomb

Thêm script cho bomb và xử lý các logic tại hàm OnTriggerEnter: tắt collider (để không va chạm nữa), chạy hiệu ứng nổ và xử lý logic nổ tại GameController.

```
1 asset usage  1 usage  NGUYEN TIEN HUNG
public class Bomb : MonoBehaviour
{
    public ParticleSystem fxExplosion;  Changed in 1 asset

    Event function  NGUYEN TIEN HUNG
    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            GetComponent<Collider>().enabled = false;
            fxExplosion.transform.SetParent( p: null);
            fxExplosion.Play();
            gameObject.SetActive(false);

            GameController.Instance.Explode();
        }
    }
}
```

Hình 4.7. Xử lý nổ trong event OnTriggerEnter của bomb

4.1.3. Hệ thống sinh quả và bomb



Hình 4.8. Component FruitSpawner

FruitSpawner có nhiệm vụ sinh và bắn các fruit, bomb từ dưới màn hình lên. FruitSpawner gồm các thuộc tính chính sau:

- **spawnArea**: là Collider, thể hiện vùng mà fruit và bomb sẽ được sinh ra. Vùng này nằm ở dưới camera (ngoài màn hình game).
- **fruitPrefabs**: danh sách prefab các fruit, dùng để sinh ra trong lúc chơi
- **bombPrefab**, **bombChance**: prefab của bomb và tỉ lệ sinh ra bomb. Thay đổi tỉ lệ sinh ra bomb cũng làm thay đổi độ khó game.
- **min/maxSpawnDelay**: thời gian delay giữa các lần sinh ra quả.
- **min/maxAngle:** góc bắn quả lên. Khi được bắn lên các fruit sẽ bay lên tới độ cao nhất và rơi theo quỹ đạo dạng parabol.
- **min/maxForce**: lực bắn fruit lên.
- **maxLifeTime**: thời gian tồn tại của fruit, bomb.

Ta có hàm **Spawn**() được gọi từ GameController, delay 1s khi game bắt đầu, sau đó bắt đầu chuỗi sinh ra các fruit và bomb liên tục theo trình tự:

1. Chọn prefab: ngẫu nhiên trong danh sách các prefab fruit, nếu đạt tỉ lệ sinh ra bomb thì đổi prefab quả thành prefab bomb.
2. Random vị trí trong vùng spawnArea, random góc xoay ban đầu, sinh ra fruit /bomb và set lifeTime cho nó (hết lifeTime sẽ bị destroy).
3. Thêm lực bắn lên trên cho fruit

4. Random thời gian delay và chờ tới lần sinh tiếp theo.

```
0+ asset usages  1 usage  NGUYEN TIEN HUNG ★ *
public IEnumerator Spawn()
{
    yield return new WaitForSeconds(1f);

    while (enabled)
    {
        // select prefab fruit / bomb
        GameObject prefab = fruitPrefabs[Random.Range(0, fruitPrefabs.Length)];
        if (Random.value < bombChance) prefab = bombPrefab;

        // random position, rotation, and instantiate
        Vector3 position = new Vector3
        {
            x = Random.Range(spawnArea.bounds.min.x, spawnArea.bounds.max.x),
            y = Random.Range(spawnArea.bounds.min.y, spawnArea.bounds.max.y),
            z = Random.Range(spawnArea.bounds.min.z, spawnArea.bounds.max.z)
        };
        Quaternion rotation = Quaternion.Euler(0f, 0f, Random.Range(minAngle, maxAngle));
        GameObject fruit = Instantiate(prefab, position, rotation);
        Destroy(fruit, maxLifetime);

        // Add upward force
        float force = Random.Range(minForce, maxForce);
        fruit.GetComponent<Rigidbody>().AddForce(fruit.transform.up * force, ForceMode.Impulse);

        yield return new WaitForSeconds(Random.Range(minSpawnDelay, maxSpawnDelay));
    }
}
}
```

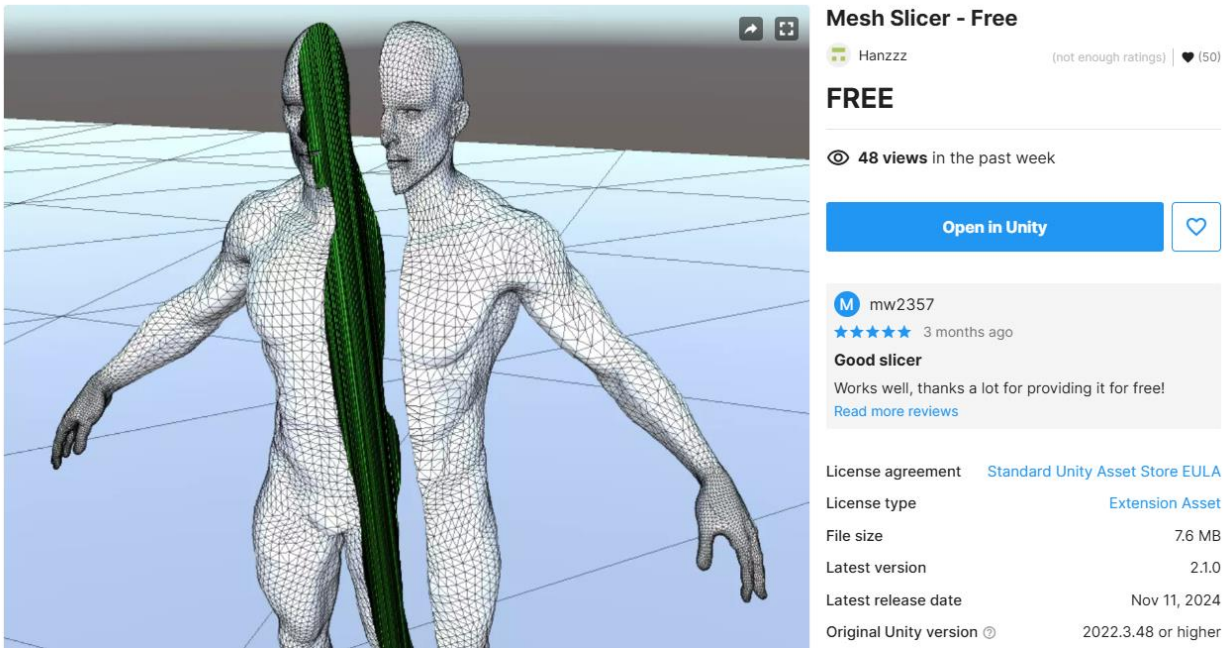
Hình 4.9. Hàm Spawn trong FruitSpawner

4.2. Xử lý logic cắt đôi quả

4.2.1. Xử lý cắt quả

Để xử lý được bài toán cắt đôi quả khi biết hướng của đường cắt, ta sử dụng 1 package bên ngoài hỗ trợ việc cho việc cắt mesh: **Mesh Slicer Free**.

Thêm package vào My Asset từ Unity Asset Store, sau đó tải và import vào dự án thông qua UPM (Unity Package Manager).



Hình 4.10. Package Mesh Slicer trên Unity Asset Store

Mesh Slicer cung cấp hàm **Slice()** cho phép cắt một đối tượng 3D (**targetGameObject**) khi biết mặt cắt (**slicePlane**), và material cho mặt cắt của 2 đối tượng mới được tạo ra (**intersectionMaterial**). Kết quả trả về là 2 GameObject tương ứng với 2 nửa được cắt từ đối tượng ban đầu nếu cắt hợp lệ (mặt cắt giao với đối tượng). Nếu cắt không hợp lệ thì trả về 2 đối tượng null.

```
3 usages  NGUYEN TIEN HUNG
public (GameObject, GameObject) Slice(GameObject targetGameObject, (Vector3,Vector3,Vector3) slicePlane, Material intersectionMaterial)
{
    Transform targetTransform = targetGameObject.transform;
    Mesh targetMesh = targetGameObject.GetComponent<MeshFilter>().sharedMesh;
    (Mesh, Mesh) slicedMesh = Slice(slicePlane, targetMesh, targetTransform, createSubmeshForIntersection: null != intersectionMaterial);
    if(null == slicedMesh.Item1)
    {
        return ((GameObject)null, (GameObject)null);
    }

    return
    (
        CreateSlicedGameObject(slicedMesh.Item1, targetGameObject, intersectionMaterial),
        CreateSlicedGameObject(slicedMesh.Item2, targetGameObject, intersectionMaterial)
    );
}
```

Hình 4.11. Hàm Slice của Mesh Slicer

Thêm script cho quả và triển khai hàm **TestSlice()**, sử dụng API Slice của package trên để test thử độ hoạt động của nó. Khai báo và sử dụng một số trường sau:

- **blade/testBlade**: mặt phẳng cắt

- **fruitModel**: model 3D của quả, sẽ là đối tượng bị cắt.
- **slicedResults**: là 2 gameObject tương ứng với 2 nửa sau khi bị cắt.
- **splitDistance**: độ dịch của 2 nửa sau khi cắt so với mặt phẳng cắt.
- **intersectionMaterial**: material cho bề mặt mới của 2 nửa sau khi bị cắt.

Như vậy đã đủ tham số để có thể sử dụng hàm và cắt model quả ra thành 2 phần. Tuy nhiên trong quá trình chơi sẽ phát sinh các trường hợp mặt cắt ở cách xa tâm của đối tượng, khi đó sẽ tạo ra 2 nửa có kích thước khá lệch nhau, 1 nửa quá to, 1 nửa quá bé tạo cảm giác chơi không tốt, vì vậy cần giới hạn lại vùng được cắt trên model quả.

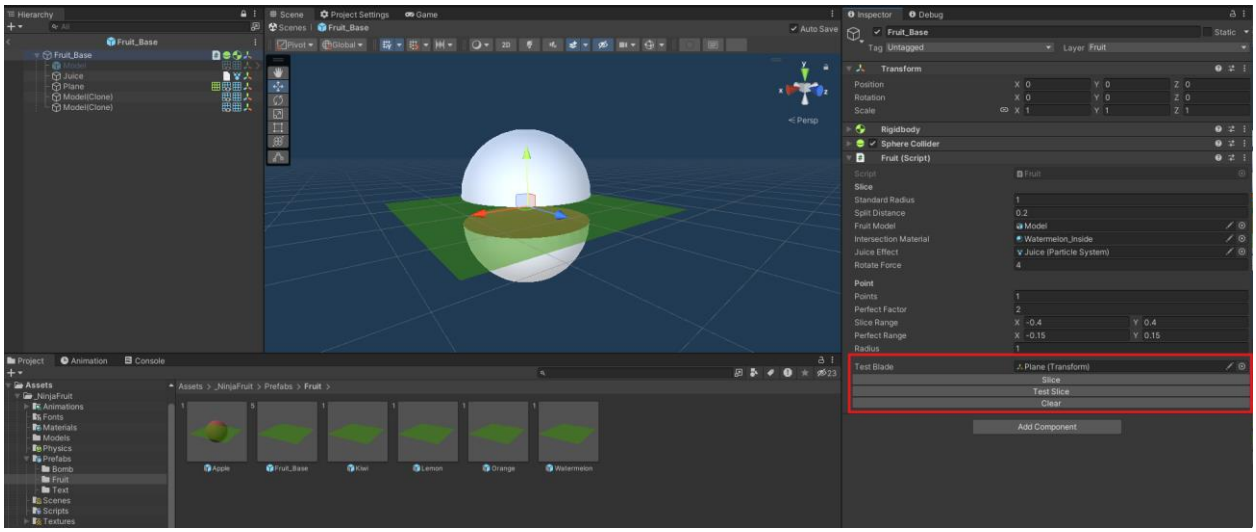
Bằng cách tính khoảng cách từ mặt phẳng cắt tới tâm của quả, và so sánh khoảng cách đó với bán kính quả, nếu nằm ngoài vùng cho phép thì dịch mặt phẳng lại biên giới hạn, nếu mặt cắt đã nằm trong vùng cho phép thì giữ nguyên vị trí mặt phẳng cắt để kết quả cắt được đúng nhất với trải nghiệm của người chơi. Triển khai thêm các trường sau:

- **standardRadius**: bán kính chuẩn của quả (Fruit_Base). Do ở mỗi quả được scale to nhỏ khác nhau nên bán kính cụ thể của mỗi quả sẽ được tính:
  radius = standardRadius * scale.
- **sliceRange**: khoảng giới hạn cắt để cho ra 2 mảnh tương đối cân bằng.

```
#region ========== TEST ==========
[Space]
public Transform testBlade;  Changed in 2 assets

[Button]
NGUYEN TIEN HUNG ★ *
public void TestSlice()
{
    ClearPreviousSlices();

    Plane slicePlane = new Plane( inNormal: testBlade.up,  inPoint: testBlade.position);
    float distanceFromPlaneToFruit = slicePlane.GetDistanceToPoint(transform.position); // Tính khoảng cách từ tâm đến mặt phẳng cắt
    float clampedDistance = Mathf.Clamp(distanceFromPlaneToFruit,  min: sliceRange.x * radius,  max: sliceRange.y * radius); // Giới hạn khoảng cách trong sliceRange

    // Nếu khoảng cách đã bị clamp, cập nhật vị trí mặt phẳng
    if (Mathf.Abs( f: clampedDistance - distanceFromPlaneToFruit) > Mathf.Epsilon)
    {
        Vector3 offset = (clampedDistance - distanceFromPlaneToFruit) * slicePlane.normal;
        slicePlane.Translate(offset); // Dịch mặt phẳng cắt lại gần tâm
    }

    slicedResults = meshSlicer.Slice(fruitModel, Get3PointsOnPlane(slicePlane), intersectionMaterial);
    if (slicedResults.Item1 == null)
    {
        Debug.Log("Slice plane does not intersect slice target.");
        return;
    }

    slicedResults.Item1.transform.SetParent(transform,  worldPositionStays: false);
    slicedResults.Item2.transform.SetParent(transform,  worldPositionStays: false);
    slicedResults.Item1.transform.position += splitDistance * testBlade.up;
    slicedResults.Item2.transform.position -= splitDistance * testBlade.up;
    fruitModel.SetActive(false);
}
#endregion
```

Hình 4.12. Hàm TestSlice cho Fruit

Trong prefab Fruit_Base, tạo một 3D Plane để làm mặt phẳng cắt test, setup các thông số và chạy thử hàm TestSlice(), ta được kết quả:



Hình 4.13. Hàm TestSlice cho Fruit

4.2.2. Xử lý các logic sau khi cắt

Sau khi đã kiểm tra hàm TestSlice() hoạt động tốt, ta triển khai các hàm chính cho đối tượng fruit. Tạo hàm **Slice()** như TestSlice() nhưng lần này blade sẽ là đối tượng tay của người chơi chứ không phải 1 mặt phẳng test nữa.

Tương tự như bomb, đối tượng fruit cũng có trigger collider và kiểm tra xem người chơi đã cắt vào quả chưa thông qua hàm OnTriggerEnter.

```
Event function  NGUYEN TIEN HUNG
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("Player"))
    {
        blade = other.GetComponent<Blade>();
        Slice();
    }
}

[Button]
1 usage  NGUYEN TIEN HUNG
public void Slice()
{
    ClearPreviousSlices();

    if (TrySlice())
    {
        HandlePostSliceEffects();
    }
}
```

```
1 usage  NGUYEN TIEN HUNG  +1
private void HandlePostSliceEffects()
{
    // tắt model cũ, tắt collider và phát hiệu ứng nước ép
    fruitModel.SetActive(false);
    fruitCollider.enabled = false;
    juiceEffect.Play();

    AlignSlices(); // xoay 2 nửa theo hướng cắt
    AddPhysicsToSlices(); // thêm vật lý vào 2 nửa quả
    HandleScore(); // xử lý điểm
}
```

Hình 4.14. Hàm xử lý va chạm trong Fruit

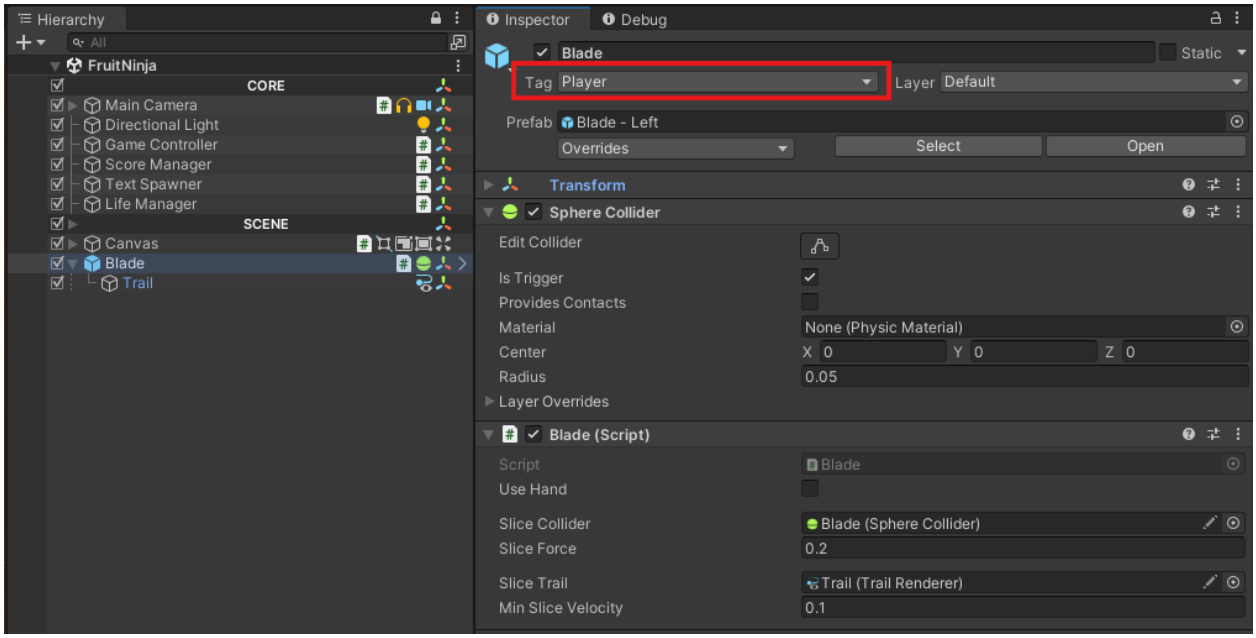
Hàm **HandlePostSliceEffects()** xử lý các logic sau khi model quả đã được cắt thành 2 nửa như:

- Tắt model quả ban đầu, chạy hiệu ứng toé nước khi bị cắt
- Xoay 2 nửa quả mới theo hướng mặt phẳng cắt
- Thêm vật lý, lực và góc xoay vào 2 nửa khi vừa mới bị cắt để 2 nửa văng nhẹ ra 2 hướng so với vị trí cắt.
- Xử lý điểm: cập nhật điểm của người chơi sau khi chém được quả.

4.3. Điều khiển chém và tích hợp Leap Motion

4.3.1. Điều khiển chém với chuột

Logic chém quả trước hết ta sẽ triển khai bằng việc sử dụng chuột. Tạo đối tượng Blade (lưỡi dao) có Collider và có tag là “Player” để có thể kiểm tra va chạm với fruit và bomb. Tạo thêm 1 Trail Renderer tạo hiệu ứng đuôi khi blade di chuyển.



Hình 4.15. Đối tượng Blade trên scene

Thêm script Blade vào đối tượng để xử lý các logic chém, bao gồm các trường sau:

- **sliceCollider**: Collider dùng để kiểm tra va chạm với fruit, bomb. Được bật lên khi đang ở trạng thái cắt (nhấn giữ chuột), và tắt đi khi đang không cắt.
- **minSliceVelocity:** vận tốc tối thiểu để nhận biết là đang cắt, khi đang nhấn giữ chuột nhưng không di chuyển chuột thì không tính là cắt, tránh trường hợp đang nhấn giữ chuột không di chuyển và fruit tự bay tới va chạm với blade vẫn tính là cắt.

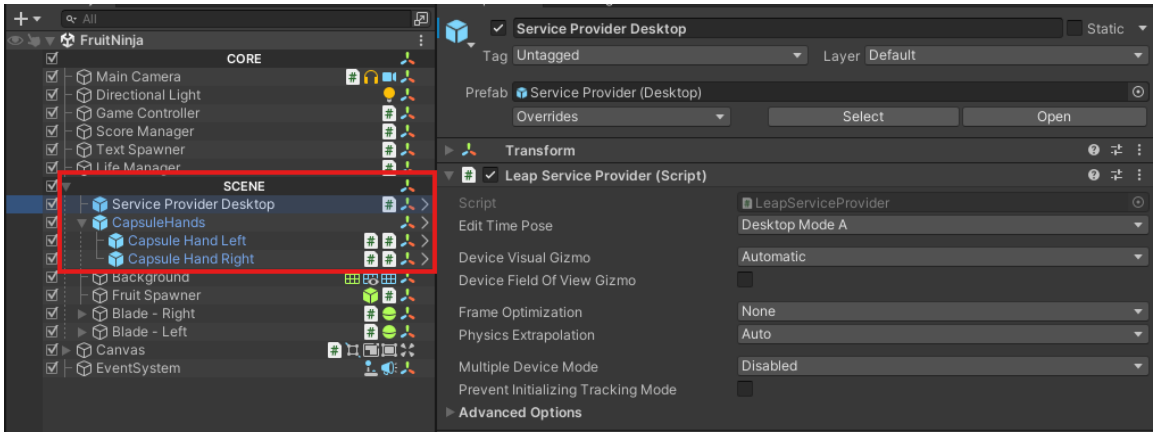
Đối với chuột, ta xử lý các logic kiểm tra để bắt đầu cắt hoặc dừng cắt trong Update. Hàm **ContinueSlice** lấy vị trí của chuột trên màn hình (**ScreenPoint**), chuyển thành vị trí trên scene (**WorldPoint**) và cập nhật vị trí của blade theo worldPoint đó.

```
#region === MOUSE ===
Event function  NGUYEN TIEN HUNG ★ *
private void Update()
{
    // dùng tay thay vì chuột
    if (useHand)
        return;

    // có bấm chuột nhưng đang ở trên UI
    if (Input.GetMouseButtonDown(0) && Utils.IsMouseOverUI())
        return;

    if (Input.GetMouseButtonDown(0)) // bắt đầu cắt
        StartSlice();
    else if (Input.GetMouseButtonUp(0)) // kết thúc cắt
        StopSlice();
    else if (slicing)
        ContinueSlice();
}

Frequently called  2 usages  NGUYEN TIEN HUNG ★
private void StartSlice(){...}

Frequently called  2 usages  NGUYEN TIEN HUNG ★
private void ContinueSlice(){...}

Frequently called  4 usages  NGUYEN TIEN HUNG ★
private void StopSlice(){...}
#endregion
```

Hình 4.16. Xử lý logic cắt quả bằng chuột

4.3.2. Điều khiển chém bằng ngón tay

Tải và cài đặt package của Ultraleap vào dự án như phần hướng dẫn trong Chương 2.2. Thêm Service Provider Desktop cùng Capsule Hands lên scene để có thể sử dụng các API từ plugin.



Hình 4.17. Setup Leap trên scene

Lúc này ta cũng phải tạo 2 đối tượng Blade trên scene đại diện cho 2 ngón tay ở bàn tay trái và phải. Sửa lại logic của script Blade, cập nhật vị trí của blade theo vị trí của ngón tay người chơi thay vì vị trí chuột trên màn hình. Thêm các trường sau:

- **leapProvider**: cung cấp data của tay theo từng khung hình (frame)
- **isUseHand**: kiểm tra đang dùng tay hay chuột để chơi
- **handType**: blade này đang là cho tay trái hay phải, khi cần cập nhật vị trí thì gọi tới tay tương ứng để lấy dữ liệu.
- **fingerType**: ngón tay dùng để cắt, blade sẽ được update vị trí theo ngón tay này.

```
Event function  NGUYEN TIEN HUNG
private void OnEnable()
{
    StopSlice();
    leapProvider.OnUpdateFrame += OnUpdateFrame;
}

Event function  NGUYEN TIEN HUNG
private void OnDisable()
{
    StopSlice();
    leapProvider.OnUpdateFrame -= OnUpdateFrame;
}

2 usages  NGUYEN TIEN HUNG
private void OnUpdateFrame(Frame frame)
{
    if (!useHand)
        return;

    hand = frame.GetHand(handType);
    if (hand != null)
    {
        if (!slicing)
            StartSlice();
        else
            ContinueSlice();
    }
    else
    {
        if (slicing)
            StopSlice();
    }
}
```

Hình 4.18. Xử lý logic cắt quả bằng hands

Khác với khi dùng chuột, ta cần đăng ký event **OnUpdateFrame** của leapProvider để xử lý dữ liệu nhận được từ thiết bị, dữ liệu này trả về dưới dạng **Frame** và chứa đầy đủ thông tin về tay, ngón tay, trạng thái, … trong khung hình hiện tại.

position = hand.fingers[(int)fingerType].TipPosition;

Vị trí của ngón tay được lấy như trên với:

- hand: tay trong khung hình hiện tại (*frame.GetHand(handType)*)
- fingers: mảng các ngón tay từ Ngón cái => trỏ => giữa => áp út => út.

- TipPosition: vị trí của đầu ngón tay

4.4. Điều chỉnh game khi tích hợp Leap Motion Controller

Một trong vấn đề lớn nhất khi tích hợp Leap Motion Controller là đối tượng Hands mà plugin cung cấp quá nhỏ so với kích thước thông thường của một project Unity. Nguyên nhân là Leap Motion sử dụng hệ tọa độ và tỷ lệ dựa trên thực tế (real-world scale). Điều này có nghĩa là kích thước bàn tay được nhận diện phản ánh kích thước thực tế của tay người (đơn vị: millimeter hoặc tương đương trong Unity là 1 unit = 1 mét).

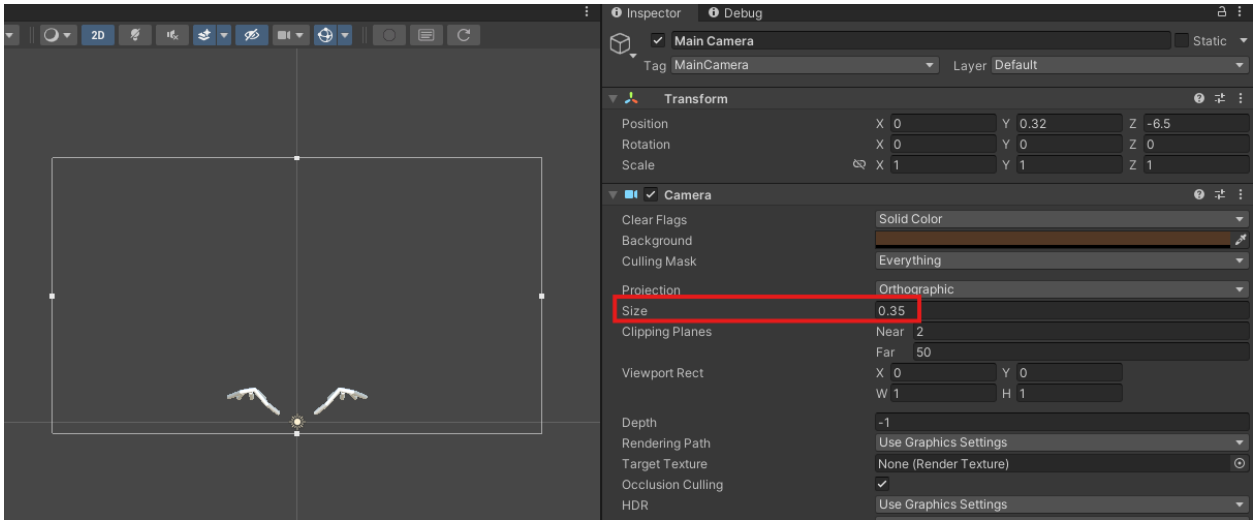
Có thể thấy với 1 unit trong unity, icon sao tương đối nhỏ chỉ với kích thước 85x85 pixel (0.85 unit) nhưng cũng đã lớn hơn rất nhiều so với kích thước của bàn tay trong Ultraleap vì một bàn tay ở thế giới thật có kích thước khoảng 15-20cm và tương đương với 0.15-0.2 unit.

Hình 4.19. Kích thước tay so với đơn vị trong Unity

Ultraleap plugin không cho phép trực tiếp scale to bàn tay vì nó mô phỏng kích thước thực của tay dựa trên dữ liệu cảm biến. Mọi cố gắng để "phóng to" mô hình tay sẽ dẫn đến việc mất đi tính chính xác khi theo dõi chuyển động.

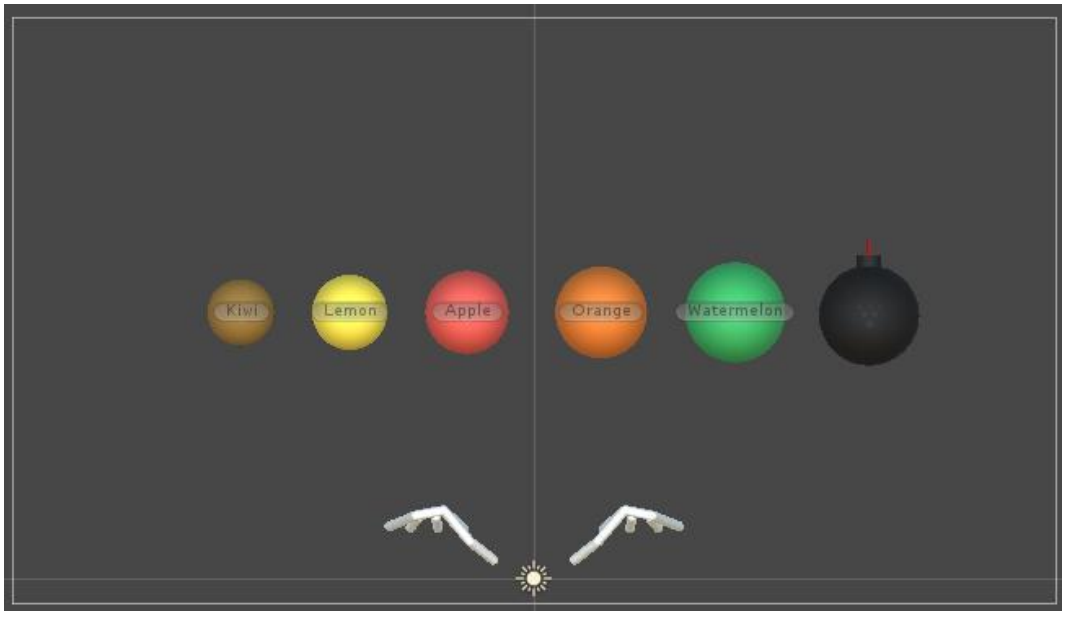
Một hạn chế nữa trong quá trình phát triển sản phẩm với thiết bị Leap Motion Controller đặt trên bàn là giới hạn phạm vi theo dõi. Với phạm vi theo dõi hẹp, chỉ trong khoảng 60cm và góc 140 độ, làm giới hạn cách người dùng di chuyển tay trong không gian 3D, nếu tay ra khỏi phạm vi theo dõi sẽ mất tín hiệu của tay.

Để giải quyết vấn các vấn đề trên, ta cần scale cả game xuống với kích thước của bàn tay làm chuẩn, đồng thời cũng cần chỉnh 1 tỉ lệ camera vừa đủ để có thể bao quát được vùng theo dõi của thiết bị, đảm bảo có thể nhìn được tay di chuyển trong toàn bộ view game.



Hình 4.20. Kích thước camera đã giảm

Các Fruit và Bomb cũng phải scale xuống một kích thước khá nhỏ.



Hình 4.21. Kích thước fruit, bomb đã giảm

Sau khi điều chỉnh các đối tượng nhỏ lại, ảnh hưởng khá nhiều đến trải nghiệm chơi, cần cân bằng thêm cả gravityScale (trọng lực) của cả game và mass (trọng lực) của từng fruit/bomb, đồng thời cũng phải giảm lực bắn của FruitSpawner lại.

4.5. Hoàn thiện game

4.5.1. Quản lý điểm và kiểm tra cắt hoàn hảo

ScoreManager (Quản lý điểm) trong game giúp:

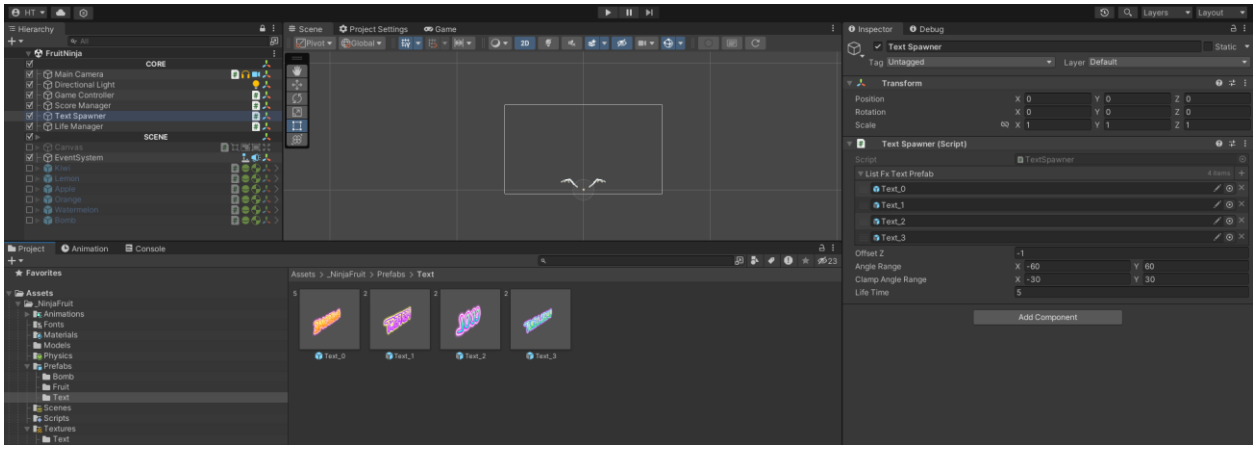
- Reset điểm khi người chơi bắt đầu 1 ván chơi mới
- Xử lý điểm mỗi khi người chơi chém vào được một Fruit
- Ghi nhận điểm cao nhất người chơi từng đạt được

Tại đối tượng Fruit, thêm thuộc tính **points** (điểm nhận được khi cắt được quả), **perfectRange** (giới hạn vùng cắt hoàn hảo), **perfectFactor** (hệ số nhân điểm khi chém hoàn hảo) và xử lý thêm logic cho điểm sau khi đã cắt quả thành 2 nửa:

```
1 usage  NGUYEN TIEN HUNG +1 *  More...
private void HandleScore()
{
    var score :int = points;

    // nếu cắt trong vùng cắt hoàn hảo (tạo 2 nửa gần bằng nhau)
    if (sliceDistance.InRange(radius * perfectRange.x, radius * perfectRange.y))
    {
        score *= perfectFactor; // nhân điểm lên
        TextSpawner.Instance.SpawnTextPerfect(transform.position, slicedResults.Item1.transform.eulerAngles.z);
    }

    ScoreManager.Instance.IncreaseScore(score);
}
```

Hình 4.22. Hàm xử lý điểm

Để lúc chém perfect được trở nên đẹp và tạo cảm giác chơi tốt hơn, tạo 1 hiệu ứng diễn các ngẫu nhiên hoạt ảnh với các text “Perfect”, “Cool”, “Excellent”, “Awesome”.



Hình 4.23. Prefab các text perfect

TextSpawner sẽ làm công việc sinh ngẫu nhiên ra 1 trong các prefab text và set vị trí, góc xoay cho text dựa vào vị trí của fruit và góc cắt. Khi vừa được sinh ra, đối tượng text sẽ tự diễn animation (scale từ bé lên và mờ dần).

Text sẽ được nghiêng cùng theo góc chém, tuy nhiên cũng có những trường hợp hướng chém gần như là dọc, nếu để text cùng góc với góc chém thì sẽ xấu. Vì vậy cần clamp lại góc chém trong 1 khoảng cho phép, nếu góc chém quá nghiêng thì coi như Text sẽ nằm ngang luôn.

```
1 usage   NGUYEN TIEN HUNG
public void SpawnTextPerfect(Vector3 position, float angle)
{
    var fxClone :GameObject = Instantiate( original: listFxTextPrefab.Rand());

    fxClone.transform.position = position + new Vector3(0, 0, offsetZ);
    angle = angle.InRange(angleRange.x, angleRange.y)
        ? Mathf.Clamp(angle, min: clampAngleRange.x, max: clampAngleRange.y)
        : 0f; // khi chém dọc quá thì coi như text ngang
    fxClone.transform.localEulerAngles = new Vector3(0, 0, angle);

    // auto destroy
    Destroy(fxClone, lifeTime);
}
```

Hình 4.24. Hàm xử lý sinh ra text

4.5.2. Hệ thống quản lý mạng chơi

Hệ thống quản lý mạng **LifeManager** cho phép người chơi có 3 mạng, mỗi khi chém vào bomb sẽ làm mất 1 mạng, đến khi hết 3 mạng (**IsOutOfLife**) thì tính là thua. Điều này giúp giới hạn thời gian chơi cũng như điểm của người chơi.

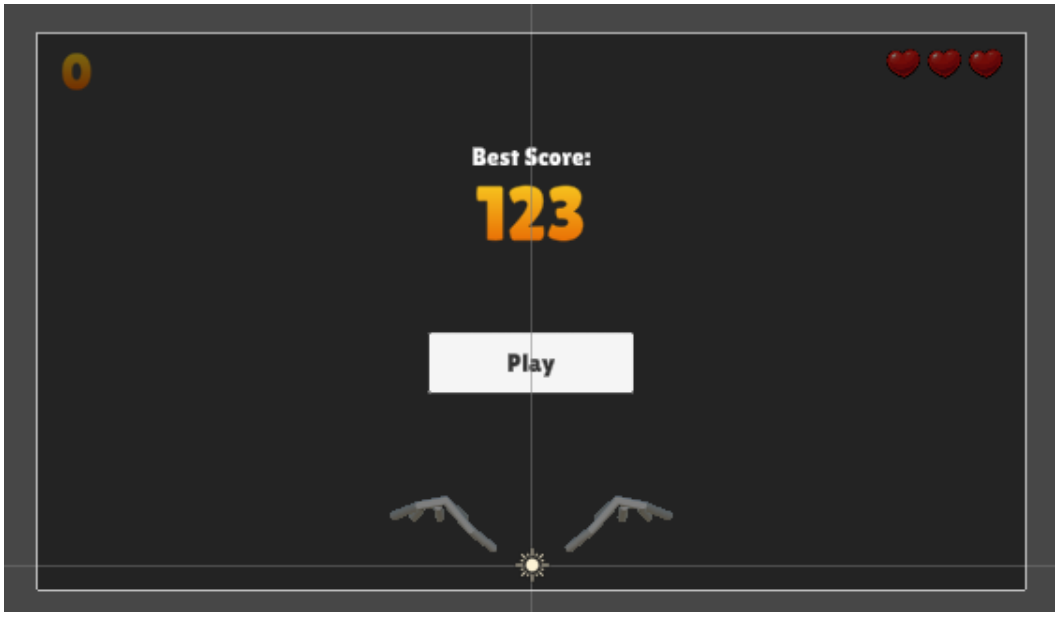
LifeManager quản lý luôn 3 icon ở UI, mỗi khi có sự thay đổi về số lượng mạng, nó sẽ cập nhật lại màu (color) của các icon đó. Màu trắng (Color.White) là khi có mạng, màu đen (Color.Black) là mạng đã mất.

```
2 asset usages  4 usages  NGUYEN TIEN HUNG  More...
public class LifeManager : MonoSingleton<LifeManager>
{
    public Image[] heartImages;  Serializable
    public int maxLife = 3;  Unchanged

    private int _life;
    3 usages  NGUYEN TIEN HUNG
    public int Life
    {
        get => _life;
        set
        {
            _life = value;
            for (var i = 0; i < maxLife; i++)
            {
                heartImages[i].color = _life > i ? Color.white : Color.black;
            }
        }
    }
    1 usage  NGUYEN TIEN HUNG
    public bool IsOutOfLife => Life <= 0;
```

Hình 4.25. Code LifeManager

4.5.3. Quản lý giao diện và quản lý game

Giao diện game chỉ đơn giản gồm 2 phần UI_Home, UI_GamePlay, và 1 ảnh trắng để tạo hiệu ứng trắng xoá màn hình khi thua do bom nổ. Tạo 1 đối tượng UI_Manager để quản lý các thành phần UI trên.



Hình 4.26. Giao diện game

Đối tượng **GameController** là thành phần điều phối chính luồng hoạt động của game. Nó thực hiện 1 số hành động chính như là:

1. Bắt đầu một màn chơi mới: chuyển giao diện Home sang giao diện chơi, reset lại điểm và mạng, bật dao cho phép chém, và bắt đầu sinh ra các fruit, bomb.

```
1 usage   NGUYEN TIEN HUNG ★   More...
public void NewGame()
{
    Time.timeScale = 1f;
    UIManager.Instance.ShowUIMainGame();
    LifeManager.Instance.ResetLife();
    ScoreManager.Instance.ResetScore();

    ClearScene();
    blade.enabled = true;
    fruitSpawner.enabled = true;
    StartCoroutine( routine: fruitSpawner.Spawn());
}
```

Hình 4.27. Hàm NewGame() trong GameController

2. Dọn dẹp scene: tìm và xoá toàn bộ các Fruit và Bomb còn sót lại trên scene, được gọi trước khi vào một màn chơi mới

```
1 usage  NGUYEN TIEN HUNG
private void ClearScene()
{
    foreach (var fruit in FindObjectsOfType<Fruit>())
        Destroy(fruit.gameObject);

    foreach (var bomb in FindObjectsOfType<Bomb>())
        Destroy(bomb.gameObject);
}
```

Hình 4.28. Hàm ClearScene() trong GameController

3. Diễn animation nổ trắng xoá màn hình và khi thua (hết mạng chơi)

```
1 usage  NGUYEN TIEN HUNG
public void Explode()
{
    LifeManager.Instance.Life--;
    cameraCtrl.ShakeCamera();

    if (LifeManager.Instance.IsOutOfLife)
    {
        blade.enabled = false;
        fruitSpawner.enabled = false;
        ScoreManager.Instance.UpdateBestScore();
        ExplodeSequence();
    }
}

1 usage  NGUYEN TIEN HUNG *
private async void ExplodeSequence()
{
    await DOVirtual.Float(1f, 0.2f, duration: 1f, onVirtualUpdate: (t :float ) => Time.timeScale = t)
        .SetEase(Ease.OutCubic).SetUpdate(true).ToUniTask(); // UniTask
    await UIManager.Instance.FadeIn();
    await UniTask.Delay(200, ignoreTimeScale: true);
    await UIManager.Instance.FadeOut();
    UIManager.Instance.OpenHome();
}
}
```

Hình 4.29. Hàm Explode() trong GameController

4.6. Kết luận chương

Như vậy chương 4 đã trình bày chi tiết quá trình xây dựng game "Chém Hoa Quả" trong Unity với tích hợp Leap Motion Controller. Từng bước, hệ thống đã được thiết kế và triển khai, bao gồm việc xây dựng các prefab cho quả và bom, logic sinh các đối tượng này, cũng như các thao tác xử lý khi quả bị chém đôi.

Bên cạnh đó, chương này cũng làm rõ cách thức điều khiển trò chơi, từ việc sử dụng chuột đến tích hợp cử chỉ ngón tay qua Leap Motion, đảm bảo sự linh hoạt và tương tác thực tế cho người chơi. Việc tối ưu hóa và điều chỉnh game để phù hợp với Leap Motion Controller cũng được thực hiện, giúp tăng cường trải nghiệm chơi game bằng cử chỉ.

Cuối cùng, các hệ thống quản lý điểm, giao diện, và cơ chế kiểm tra đã được hoàn thiện, đảm bảo phiên bản prototype của trò chơi vận hành ổn định. Những nội dung này tạo cơ sở quan trọng để đánh giá hiệu quả ứng dụng công nghệ Leap Motion trong việc thiết kế và lập trình game, đồng thời hướng tới các bước cải thiện và phát triển trong tương lai.

KẾT LUẬN ĐỒ ÁN

Qua quá trình nghiên cứu và thực hiện đồ án, cá nhân tác giả đã tích lũy được nhiều kiến thức và kỹ năng quan trọng trong lĩnh vực phát triển game. Từ việc tìm hiểu sâu về công nghệ nhận diện cử chỉ tay, tích hợp Leap Motion với Unity, đến việc thiết kế và xây dựng game thực tế, tác giả đã trải nghiệm toàn bộ quy trình phát triển một sản phẩm game sáng tạo và độc đáo.

Kết quả đạt được:

- Hiểu rõ về công nghệ nhận diện cử chỉ tay và thiết bị Leap Motion Controller và các ứng dụng trong thực tiễn..
- Có kiến thức về Unity Engine, Ultraleap plugin và cách tích hợp Leap Motion trong Unity.
- Tích hợp thành công Leap Motion với Unity, phát triển một trò chơi nguyên mẫu (prototype) hoạt động.

Hạn chế:

- Game chưa hoàn thiện: do thiếu sót tài nguyên từ artist, model 3D cũng như 1 bản thiết kế game hoàn chỉnh nên game mới chỉ dừng ở mức prototype và demo được cơ chế chính dùng tay để tương tác.
- Về thiết bị: Leap Motion Controller phù hợp để tích hợp vào các thiết bị VR-AR hơn là phát triển 1 ứng dụng để bàn do giới hạn về khoảng cách cũng như góc theo dõi.
- Tích hợp Unity: Ultraleap plugin lấy đơn vị đo thực tế để cung cấp dữ liệu cho các tay trong game và các kích thước này rất bé so với các đơn vị trong dự án game bình thường. Điều này dẫn tới game phải scale nhỏ theo và điều chỉnh nhiều thông số vật lý khác.

Hướng phát triển đề tài:

- Hoàn thiện đồ hoạ, âm thanh, model các quả, hiệu ứng và trail của đường chém.
- Phát triển thêm các mode chơi mới xoay quanh cơ chế dùng tay để chém.
- Phát triển thêm một số mini game nhỏ, ứng dụng các tính năng khác như cầm nắm, tạo dáng bàn tay của Leap Motion.

DANH MỤC TÀI LIỆU THAM KHẢO

[1] Leap Motion: https://en.wikipedia.org/wiki/Leap_Motion

[2] Leap Motion Controller:
http://arduino.vn/video/1485-leap-motion-controller-review-tren-tay-nhanh-leap-motion

[3] Chi tiết thông số Leap Motion Controller:
https://www.ultraleap.com/datasheets/Leap_Motion_Controller_Datasheet.pdf?_gl=1*aq5w96*_ga*MTM2MDY1MDE2Ni4xNzI4Mjg0OTcz*_ga_5G8B19JLWG*MTczNDI1ODA4Ny4xNi4xLjE3MzQyNTgxMzcuMTAuMC4w

[4] Game The Unspoken: https://insomniac.games/game/the-unspoken/

[5] Unity Engine: https://en.wikipedia.org/wiki/Unity_(game_engine)

[6] Ultraleap in Unity: https://docs.ultraleap.com/xr-and-tabletop/xr/unity/index.html

[7] Fruit Ninja: https://vi.wikipedia.org/wiki/Fruit_Ninja