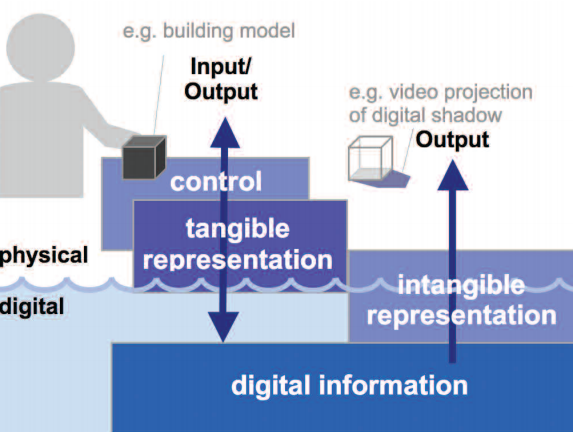
## 1 Giới thiệu

## 2 Các nghiên cứu liên quan

### 2.1 Tangible là gì?

Tangible user interface (TUI) là giao diện người dùng giúp con người có thể tương tác với thông tin kĩ thuật số thông qua môi trường vật lí. Mục đích của sự phát triển TUI là để tăng cường tính cộng tác, học hỏi, thiết kế bằng cách vật lí hóa những thông tin kĩ thuật số ví dụ như cầm nắm đồ vật, vật liệu [1].

Hình 1. Tương tác có tính chất tangible và không tangible [1]

Theo như Hình 1, môi trường hỗ trợ tương tác Tangible là đưa yếu tố tương tác vật lý vào một môi trường chỉ hỗ trợ trình chiếu đơn thuần, đồng hộ hóa hai yếu tố này lại với nhau. Môi trường tương tác có yếu tố tangible giúp người dùng nắm bắt thông tin một cách trực tiếp, hiệu quả hơn do sự phản hồi xúc giác trong quá trình tương tác mang lại [2].

Tangible có thể ứng dụng vào các môi trường tương tác trong nhiều lĩnh vực khác nhau

Tangible (techniques and devices): Bắt đầu từ những hệ thống đơn giản từ cuối thế kỉ 20 như hệ thống PingPongPlus của Hiroshi Ishii và các cộng sự [1] dùng để số hóa trò chơi Ping Pong cổ điển, tác giả đã dùng một mặt bàn tương tác kết hợp hệ thống cảm ứng va chạm được tích hợp vào bàn và công nghệ chiếu ảnh. Các thiết bị Tangible ngày càng phát triển với nhiều kĩ thuật mới hơn được tích hợp vào như: giao diện tabletop tangible sử dụng nền tảng theo dõi đối tượng  (Object tracking platform), được đề xuất từ hệ thống Sensetable có thể theo dõi vị trí và hướng của nhiều vật có kết nối không dây trên một mặt bàn hiển thị của James Patten và cộng sự [2], cho thiết bị điều khiển âm nhạc điện tử của nhóm Ben Recht [3]. Thêm vào đó sự gia tăng băng thông rộng và kết nối Internet tốc độ cao cũng hỗ trợ cho việc mở rộng các thiết bị Tangible, đơn cử là hệ thống Kinected Conference của Anthony DeVincenzi và các cộng sự [4] dùng để tăng cường hiệu quả giao tiếp trong vấn đề Hội thoại Video (Video Conference) bằng cách tạo ra các hình ảnh ba chiều giàu ngữ nghĩa chứa thông tin về thành phần vật chất và vị trí của chúng. Sự phát triển trong khoa học vật liệu cũng là một tiền đề để nâng cao khả năng của các thiết bị Tangible, dự án PneUI của Lining Yao [5] giới thiệu công nghệ cho phép xây dựng những thiết bị mà bề mặt tương tác có thể thay đổi hình dáng bằng các vật liệu mềm được nén khí.

- Một số ứng dụng tiêu biểu:

- Giáo dục: [][] làm về ứng dụng….

- Urban plannning.

- [][]

### 2.2 Các kỹ thuật tương tác trong môi trường hỗ trợ tương tác tangible

Hệ thống sử dụng

### 2.3 Hiển thị thông tin trong môi trường hỗ trợ tương tác tangible

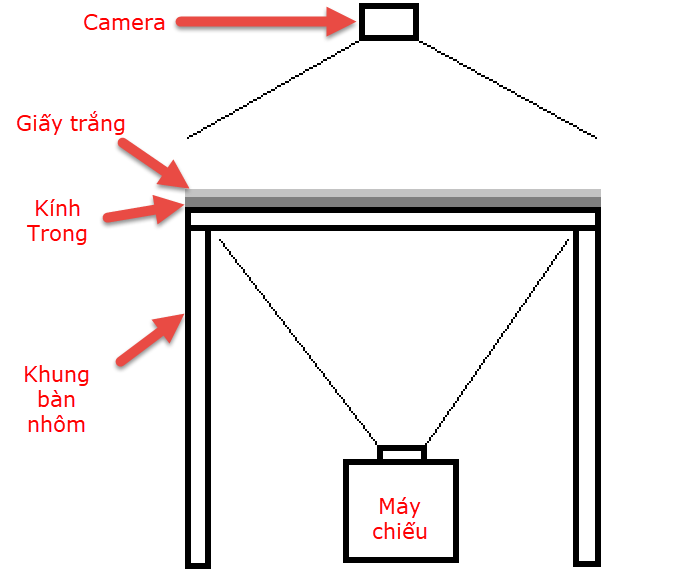
## 3 Hệ thống xây dựng

### 3.1 Cấu hình hệ thống

Hệ thống của chúng em bao gồm :

Một khung bàn nhôm có chiều rộng 45cm và chiều dài là 73cm, bàn cao 103cm. Mặt bàn chỉ có viền cạnh và rỗng bên trong. Đặt phía trên khung là một tấm kính trong suốt và phía trên tấm kính là giấy trắng để hứng ảnh (Hình 2).

Máy chiếu Sanyo PLC-XW60 với cường độ sáng 2000 Ansi Lumens, độ phân giải 1024x768, kích thước hình ảnh từ 80 đến 300 inches. Máy chiếu theo góc 90 độ hướng lên từ dưới chân bàn lên mặt kính trong, hình ảnh sẽ được giữ lại ở mặt giấy trắng, để hình ảnh thuận chiều với mắt người nhìn từ phía trên thì chúng em tiến hành lật màn hình phát ra từ máy chiếu theo chiều dọc. Với máy chiếu trên có cảm biến về góc chiếu nên chúng em đã điều chỉnh keystone để hình ảnh phát ra là hình chữ nhật với góc người nhìn, cuối cùng là điều chỉnh độ thu phóng và focus của thấu kính.

Camera Logitech C170 có khả năng quay video 1024x768, độ phân giải 5 megapixel, USB 2.0 để thu hình từ phía trên. Camera được lắp cao cách 75cm so với mặt bàn, có góc nhìn sao cho lấy đủ mặt bàn và hình ảnh hứng được. Camera được dùng để thu ảnh và nhận diện vật thể trong quá trình sử dụng. Để hình ảnh phát ra hứng được là rõ nhất thì ngoài việc chỉnh độ focus của thấu kính chúng em có 2 hướng giải quyết: Một là giảm ánh sáng môi trường bên ngoài tác động vào mặt bàn – Đây là cách giải quyết chúng em đang thực hiện – với mức sáng hiện tại chúng em đang sử dụng là 4 lux trở xuống (ánh sáng phòng thí nghiệm ban ngày không mở đèn phòng và đóng rèm là 8 lux). Hai là thay đổi chất liệu giấy hứng ảnh để thu được hình ảnh tốt hơn, nhưng hiện tại chúng em chưa tìm ra được chất liệu giấy phù hợp. – Thiết bị di động để hỗ trợ tương tác trong quá trình hoạt động của hệ thống. Ở đây chúng em sử dụng điện thoại Xiaomi có màn hình 5.5 inches Full HD, hệ điều hành Android 7.0, CPU Snapdragon 625 8 nhân 2.0 Hz, RAM 3 GB, chip đồ họa Adreno 506 và máy tính bản Huawei Mediapad T3 màn hình 8 inches, độ phân giải 1280x800 pixels, hệ điều hành Android 7.0, CPU Qualcomm MSM8917 1.4 Hz, RAM 2 GB, chịp đồ họa Adreno 308. (Hình 3)

Hình 3 Hình ảnh thực tế hệ thống



Hình 2 Sơ đồ bố trí hệ thống

### 3.2 Cách hệ thống làm việc

#### 3.2.1 Nhận dạng vật thể tương tác

Việc nhận dạng chính xác vật thể tương tác đóng vai trò quan trọng trong ứng dụng vì nó giúp người dùng giao tiếp được với hệ thống và nâng cao trải nghiệm của người dùng. Để có thể triển khai nhanh ứng dụng với chi phí rẻ, chúng em quyết định dùng một camera màu hỗ trợ FullHD (webcam) hiệu …. để thực hiện lấy ảnh màu cho việc nhận dạng.

Chúng em áp dụng các phương pháp xử lý ảnh cơ bản để nhận dạng vật theo màu sắc theo quy trình sau:

Lấy mẫu ảnh

Tọa độ vật thể

Chuyển sang hệ HSV

Tìm vùng chứa vật thể

Lọc nhiễu

Tracking bằng Camshift

Video

Sơ đồ 1: Quy trình nhận dạng vật thể bằng xử lý ảnh

Sơ đồ 1 mô tả quy trình nhận dạng một vật thể theo kênh màu. Bước đầu tiên ta sẽ lấy vùng ảnh chứa vật thể cần nhận dạng, sau đó chuyển ảnh sang hệ màu HSV rồi tiến hành tính toán biểu đồ Histogram của vùng màu được chọn. Ta bắt đầu tìm vùng ảnh chứa vật thể cần nhận dạng. Dựa vào biểu đồ histogram, ta áp dụng phương pháp Back Projection để lấy được ảnh xác suất màu sắc có thể xuất hiện trong mỗi điểm ảnh như Hình 2. B. Sau đó đem chồng lên 1 lớp ảnh Mask (Hình 2. A) đã lọc theo ngưỡng màu từ ảnh gốc. Kết quả thu được ta sẽ thu được một ảnh trắng đen như Hình 2. C với vùng màu trắng là vùng có kênh màu cần nhận dạng. Tiến hành thí nghiệm nhận dạng một nắp chai màu xanh, ta thấy ảnh thu được sau khi tìm vùng chứa vật thể còn nhiễu khá nhiều, nguyên nhân là do các vùng lân cận có thể có kênh màu gần giống với vật thể. Để thu được kết quả chính xác hơn, ta tiến hành bước lọc nhiễu.



A

B

C

D

Hình 4 : Kết quả các bước trong quy trình nhận dạng vật thể

Để lọc các giá trị nhiễu này, ta áp dụng phương pháp lọc nhiễu hình thái (morphology) với 2 thao tác cơ bản là co ảnh (erosion) và giãn nở ảnh (dilation). Ý tưởng chính của phương pháp này là dùng phép toán erosion làm giảm kích thước của đối tượng, tách rời các đối tượng gần nhau, từ đó làm tiêu biến các điểm nhiễu nhỏ. Sau đó sẽ dùng phép toán dilation làm cho đối tượng ban đầu trong ảnh tăng lên về kích thước để vùng biên của vật thể tròn trịa, mịn hơn.

Toán tử erosion :



Trong đó:

* A: Ma trận điểm ảnh của ảnh nhị phân.
* B: Là phần tử cấu trúc.

Phép co ảnh erosion sẽ cho ra một tập điểm ảnh c thuộc A, nếu ta đi chuyển phần tử cấu trúc B theo c, thì B nằm trong đối tượng A.

Một ví dụ trực quan hơn, ta có ma trận điểm ảnh Isrc, ma trận điểm ảnh sau phép co Idst và cấu trúc phần tử B. Ứng với công thức ở trên, ta lần lượt đặt phần tử cấu trúc vào các điểm ảnh có giá trị 1 của ma trận điểm ảnh Isrc. Kết quả thu được là ma trận điểm ảnh Idst.

Hình 5 nhgnh

Toán tử dilation : với phương pháp tương tự như toán tử erosion, nhưng cấu trúc phần tử B sẽ mở rộng vùng chứa ô có giá trị

Sau quá trình lọc nhiễu, ta thu được Hình 2. D để tiến hành việc truy vết ảnh. Ở đây chúng em dùng thuật toán Camshift (Continuously Adaptive Meanshift) trong OpenCV. Thuật toán này bản chất là áp dụng thuật toán Meanshift lên mỗi frame ảnh của nguồn video thu được từ webcame.

#### 3.2.2 Nhận dạng vật thể tương tác bằng Vuforia trong Unity

Với mục đích sử dụng Vuforia để nhận diện Object trong trường hợp vật cần nhận diện có nhiều họa tiết hoa văn chứ không phải là một mặt phẳng đơn sắc. Chúng em tiến hành sử dụng Vuforia để kiểm tra mức độ hiệu quả của nó so với quy trình nhận dạng .

Đăng kí tài khoản sử dụng Vuforia API :

Đầu tiên, truy cập vào <https://developer.vuforia.com/vui/auth/register> để đăng kí tài khoản developer (Hình 6). Sau khi điền đủ thông tin vào mẫu đăng kí và xác nhận tạo tài khoản, ta xác thực việc đăng kí với email được gửi đến.

Hình 6 Đăng kí tài khoản Vuforia

Sau khi xác nhận tài khoản, ta đăng nhập vào hệ thống với đường dẫn: <https://developer.vuforia.com/vui/auth/login>

Sử dụng Vuforia API với Unity :

Trước hết, ta cần tải bộ cài từ trang Download của Vuforia với đường dẫn sau: <https://developer.vuforia.com/downloads/sdk>. Cài đặt được tải ở dạng là một package của Unity, khi tải xong sẽ được cài trực tiếp vào Unity. Tuy nhiên đây là cách cài đặt dành cho Unity có phiên bản trước 2017.2, từ phiên bản 2017.2 trở về sau, Vuforia có thể được cài đặt với các package khác khi cài đặt Unity.

Tiếp theo, ta khởi động Unity, Chọn Edit => Project Settings => Player. Ở cửa sổ Inspector, ta đánh dấu vào “Vuforia Augmented Reality” (Hình 7).

Hình 7 Cho phép sử dụng Vuforia trong project

Để sử dụng các chức năng của Vuforia thì ta chọn Game Object => Vuforia rồi chọn chức năng tương ứng, ở đây chúng em lấy chức năng cơ bản của Vuforia là Image Target. Để sử dụng Image Target thì ở trang developer của Vuforia ta cần tạo một App Key để sử dụng. Khi điền đủ thông tin và xác nhận tạo key, ta có một App Key được tạo như hình dưới, ta nhấn vào App Key Name để lấy Key (Hình 8).

Hình 8 Giao diện lấy App Key

Ta copy phần App Key, ở màn hình Unity, ta chọn Window => Vuforia Configuration, sau đó Key vào ô App License Key (Hình 9).

Hình 9 Gắn App Key vào project

Bây giờ, ta đã có thể sử dụng Image Target, chức năng cơ bản của Image Target là nhận diện một hình ảnh 2D đã được thêm vào database của Vuforia. Để thêm một hình ảnh vào data base ta đăng nhập vào trang chủ Vuforia, ở Tab Develop chọn Target Manager và nhấn Add Database. Đặt tên Database và tạo với type là Device.

Hình 10 Tạo Database hình ảnh nhận diện



Ta nhấn vào tên Database và nhấn Add Target để thêm dữ liệu vào Database. Có thể thêm hình ảnh 2D hoặc các vật thể 3D (Hình 10). Sau khi upload dữ liệu, điền kích thước và đặt tên thì ta nhấn Add để xác nhận. Khi dữ liệu được đưa vào Database thì mức độ đánh giá khả năng nhận diện được thể hiện qua phần Rating. Cuối cùng nhấn vào Download Database (All) để tải về Unity. Khi tải về ta chọn Development platform là Unity Editor rồi nhấn Download . File tải về dưới dạng là package của Unity sẽ được cài trực tiếp (Hình 11).

Ở giao diện Unity, ta chọn Database tương ứng vừa tải về và chọn Image Target là ảnh được đưa vào Database (Hình 12). Sau đó là có thể Start Game Mode của Unity và nhận diện.

Hình 12 Chọn Target theo hình ảnh từ Database



Hình 11 Tải Database về project

* Tiêu chí Rating ảnh của Vuforia và các yếu tố ảnh hưởng đến việc nhận diện



Hình 13 Dựa vào độ tương phản - Nguồn : <https://www.edrawsoft.com/umldiagramtemplate.php>

Vuforia có thể nhận diện và theo dấu Targets bằng việc phân tích sự tương phản dựa trên những đặc tính được rút trích từ Target đó thể hiện cho camera như họa tiết của ảnh, tỉ lệ ảnh, chất lượng ảnh, độ cân bằng trong phân bố các chi tiết ảnh… Có thể cải thiện hiệu quả nhận diện bằng cách cải thiện những đặc tính đó (tăng độ chi tiết, độ tương phản màu sắc tốt (), không bị lặp nội dung, chi tiết có nhiều góc cạnh). Format của ảnh phải là ảnh PNG hoặc JPG 8 hoặc 24 bit, nhỏ hơn 2MB, với ảnh JPG phải dùng hệ màu RGB hoặc greyscale (không sử dụng hệ màu CMYK). Để đánh giá tiêu chí Rating với ảnh Greyscale thì ta dựa vào Greyscale Histogram, ảnh có độ tương phản cao, có Histogram rộng và phẳng thường là ảnh có Rating cao (H).



Hình 14 Độ tương phản ảnh Grayscale tương ứng với histogram - Nguồn : <https://www.edrawsoft.com/umldiagramtemplate.php>

Về camera thì nếu có chế độ focus sẽ tốt hơn cho việc nhận diện. Bên cạnh đó, điều kiện ánh sáng môi trường cũng có ảnh hưởng lớn đến việc nhận diện và theo dấu. Vuforia hoạt động tốt nhất với ánh sáng trong phòng vì nó ổn định và dễ dàng điều chỉnh cường độ sáng. Kích thước Target tốt nhất từ 5 inches hoặc 12cm trở lên về chiều rộng cùng với kích thước hợp lí theo tỉ lệ ảnh về chiều cao. Khoảng cách từ ảnh đến camera trong điều kiện hoàn hảo khoảng bằng 10 lần chiều rộng của ảnh. Ảnh được in ra cần đảm bảo độ phẳng, không bị bẻ cong hay nhăn, nên sử dụng giấy bìa cứng để in ảnh nhận diện. Bề mặt ảnh với camera nên là bề mặt nhám, vì bề mặt bóng dưới nguồn sáng và góc chiếu sáng ngẫu nhiên có thể dẫn đến hiện tượng phản xạ, khúc xạ làm mất đi chi tiết của ảnh. Góc nhìn của camera đến ảnh nên trực diện và tránh góc nhìn quá hẹp so với bề mặt ảnh nhận diện.

Sau quá trình thử nghiệm với Vuforia, chúng em thấy được lợi điểm của nó cho phép nhận diện và theo dấu được vật có bề mặt phức tạp. Tuy nhiên, nó cũng bị phụ thuộc vào điều kiện môi trường như quy trình ở Sơ đồ 1 và khả năng mapping tọa độ thu được của vật trong thế giới ảo với vật trong thế giới thật.

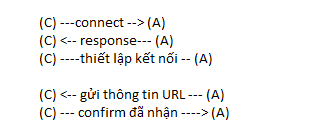
#### 3.2.3 Giao tiếp qua giữa các thiết bị tương tác

Theo như kiến trúc ứng dụng trên được trình như ở …., ứng dụng chia ra thành các module độc lập với nhau, mỗi module chịu một trách nhiệm riêng biệt như: module xử lý dữ liệu và render hình ảnh (A), module nhận dạng vật thể (B), module tương tác nội dung với người dùng như điện thoại, máy tính bảng (C). Thêm vào đó, các module cần một phương thức để giao tiếp, truyền đạt dữ liệu qua lại lẫn nhau một cách hiệu quả.

Trong giới hạn luận văn này, ứng dụng được triển khai trên môi trường mạng cục bộ LAN, nên chúng em quyết định dùng phương thức Socket để giao tiếp giữa các module. Socket là một phương pháp để thiết lập kết nối truyền thông giữa một chương trình yêu cầu dịch vụ (client) và một chương trình cung cấp dịch vụ (server) trên mạng LAN, WAN hay Internet.

Module (A) và (B) chạy cùng trên một máy tính và (B) đóng vai trò là client, chỉ có nhiệm vụ gửi tọa độ truy vết được cho (A) xử lý. (B) được viết bằng ngôn ngữ Python và dùng thư viện OpenCV để hỗ trợ việc truy vết. (A) đóng vai trò là server với (B) và chỉ tiếp nhận dữ liệu từ (B) nếu có và không có phản hồi lại. (A) sẽ lấy dữ liệu tọa độ truy vết được từ (B) để tiến hành căn chỉnh tọa độ phục vụ cho việc render hình ảnh

Module (A) và (C) chạy trên 2 thiết bị khác nhau và (A) đóng vai trò là server, (C) là client, giao tiếp với nhau theo kịch bản



Đối với hệ điều hành Android, để sử dụng SocketIO, ta thêm thư viện vào file gradle như ở Mã 1.

|  |
| --- |
| import java.net.Socket |

Mã 1

Để mở một kết nối Socket, ta thực hiện đoạn lệnh sau :

**Socket** mSocket = **new** **Socket** (mIPAddress, mPort);

Để gửi gói tin từ module C về module A, ta sử dụng đoạn lệnh

**PrintWrite** mSender = **new** **PrintWriter** (socket.getOutputStream(), true);

mSender.print(“Hello server”);

Để nhận c

Đối với Module B, trong ngôn ngữ Python, để sử dụng SocketIO, ta chỉ cần thêm thư viện socket và khởi tạo một đối tượng socket

import socket

socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

và thực hiện việc gửi dữ liệu như sau

sock.sendto(data, (IP, PORT))

Ở đây, module (B) chỉ có nhiệm vụ gửi vị trí tọa độ vật thể truy vết được 1 cách liên tục mà không cần quan tâm việc phản hồi, nên chúng em đã dùng phương thức gửi UDP, cụ thể là dùng tham số socket.SOCK\_DGRAM để thiết lập việc gửi theo hình thức UDP. Phương thức gửi UDP tuy không thể xác nhận việc gửi có thành công hay không, và giới hạn kích thước gói tin gửi bị giới hạn, tuy nhiên tốc độ gửi sẽ nhanh hơn nhiều so với TCP. Ở ngữ cảnh này, việc dùng UDP là hợp lý

Đối với môi trường Unity, để sử dụng SocketIO, ta làm theo các bước sau

#### 3.2.3 Kiến trúc hệ thống

Hệ thống sự kiện (Event System) được thiết kế với mục đích quản lí kịch bản hoạt động của hệ thống một cách hiệu quả và tách biệt các thành phần của hệ thống với nhau để chúng hoạt động độc lập và dễ dàng thay đổi, mở rộng. Các thành phần ở đây là việc xử lí một sự kiện xảy ra, và việc kiểm tra điều kiện để kích hoạt sự kiện đó. Nó cho phép việc load nhanh một kịch bản và có thể thay đổi kịch bản đó một cách linh hoạt mà không phụ thuộc vào kiến trúc xử lí sự kiện bên trong. Bên cạnh đó hệ thống sự kiện cho phép việc kích hoạt liên hoàn sự kiện một cách dễ dàng. Hệ thống sự kiện không chỉ có kịch bản sự kiện ẩn liên hoàn mà còn bao gồm cả các sự kiện và việc xử lí sự kiện do người dùng tạo ra (như cách xử lí sự kiện thông thường). Ở đây chúng em tập trung vào việc xử lí liên hoàn sự kiện theo kịch bản. Minh họa hệ thống sự kiện như Sơ đồ 2:

Sơ đồ 2 Hệ thống sự kiện

##### 3.2.3.1 Thành phần của hệ thống sự kiện liên hoàn

Sơ đồ 3 Hệ thống sự kiện liên hoàn



Một mảng các đối tượng Event, mỗi Event tương ứng với một bộ các hành động tác động lên các Object tùy thuộc vào kịch bản.

Một Dictionary chứa các cờ hiệu (Flags), trong quá trình hoạt động, chương trình sẽ duyệt Dictionary này một cách xuyên suốt. Nếu có một tổ hợp cờ nào phù hợp với một Event tương ứng thì Event đó sẽ được kích hoạt.

Một Dictionary các Object sẽ bị tác động trong kịch bản, ở đầu chương trình sẽ tiến hành đăng kí các Object này vào Dictionary.

Một kịch bản đầu vào ở dạng JSON File, mô tả các Event. Cấu trúc gồm tên của Event, Action, Pre-conditions, Post-conditions.s

##### 3.2.3.2 Cấu trúc của một Event

Sơ đồ 3 mô tả mối quan hệ giữa các thành phần trong hệ thống Event, chi tiết các thành phần như sau :

Pre-conditions: Tập hợp các điều kiện đầu vào, một Pre-condition gồm các Conditions. Sự kiện được kích hoạt khi một Pre-condition thỏa.

Conditions: Là một chuỗi các điều kiện được nối với nhau bằng toán tử &&, một Pre-condition được kích hoạt khi chuỗi các điều kiện đó đều thỏa.

Comparators: Là các toán tử so sánh dùng để so sánh các cờ hiệu với một giá trị nào đó, nhằm kích hoạt Conditions từ đó kích hoạt Pre-condition và cuối cùng là kích hoạt sự kiện. Các toán tử này bao gồm: Equal, GreaterThan, LessThan, LessThanOrEqual, GreaterThanOrEqual. Các toán tử này kế thừa Abstract Class Comparator và được tạo ra một lần duy nhất ở đầu chương trình để sử dụng (Sơ đồ 4). Nói cách khác đây là một Global Helper Class.

Sơ đồ 4 Sơ đồ lớp của Comparator

public class GreaterThanOrEqual : Comparator

{

public override bool compare(string FlagId, int value)

{

return Global.getFlag(FlagId) >= value;

}

}

Mã 2 Minh họa một Comparator

Action: Là một hành động sẽ tác động lên một GameObject trong hệ thống khi sự kiện được kích hoạt (ví dụ như ẩn, hiện, di chuyển vị trí, thu nhỏ, phóng to,…). Các class này kế thừa class Action Type là Base Abstract Class cho các Action này (Sơ đồ 5)

Mã 3 Minh họa một Action

public class RotateObject : ActionType

{

public override void DoAction(GameObject gameObject, ParamsRaw paramsRaw)

{

gameObject.transform.Rotate(0, 0, 30 \* Time.deltaTime, Space.World);

}

}

Post-actions: Là chuỗi các hành động dùng để cập nhật các phần tử của Dictionary Flags sau khi Action của Event được thực hiện. Đây là cơ chể để thực hiện việc kích hoạt liên hoàn sự kiện trong hệ thống. Vì Dictionary Flags sẽ luôn được thăm qua mỗi lần lặp Update, dựa vào trạng thái cờ mà sẽ kích hoạt sự kiện, nếu một sự kiện được kích hoạt, sau đó Post-actions cập nhật lại Dictionary Flags tạo ra một tổ hợp trạng thái mới kích hoạt một sự kiện khác, từ đó tạo ra hiệu ứng liên hoàn.

Sơ đồ 5 Sơ đồ lớp của Action

Operator: Mỗi hành động trong Post-actions là một Operator (cộng, trừ, nhân, chia, gán,…) tác động đến giá trị của Dictionary Flags (Sơ đồ 6).

public class Mod : Operator

{

public override int Operate(string FlagId, int value)

{

return Global.setFlag(FlagId, Global.getFlag(FlagId) % value);

}

}

Mã 4 Minh họa một Operator

Sơ đồ 6 Sơ đồ lớp của Operator

#### 3.2.4 Kĩ thuật Calibration để mapping tọa độ

Chúng em dùng quy trình nhận dạng vật thể theo Sơ đồ 1 để nhận diện và theo dấu object có màu đơn sắc. Sau khi tọa độ tâm của vật thể đó trên bức ảnh theo đơn vị pixel, góc tọa độ của Frame ảnh đó ở góc trái trên. Nhưng ở Unity, để làm việc thì tọa độ tương đối giữa các Object được tính từ tâm của Object này đến Object kia, chính vì thế cần có một công thức để map tọa độ nhận được sang tọa độ cần xử lí ở Unity.

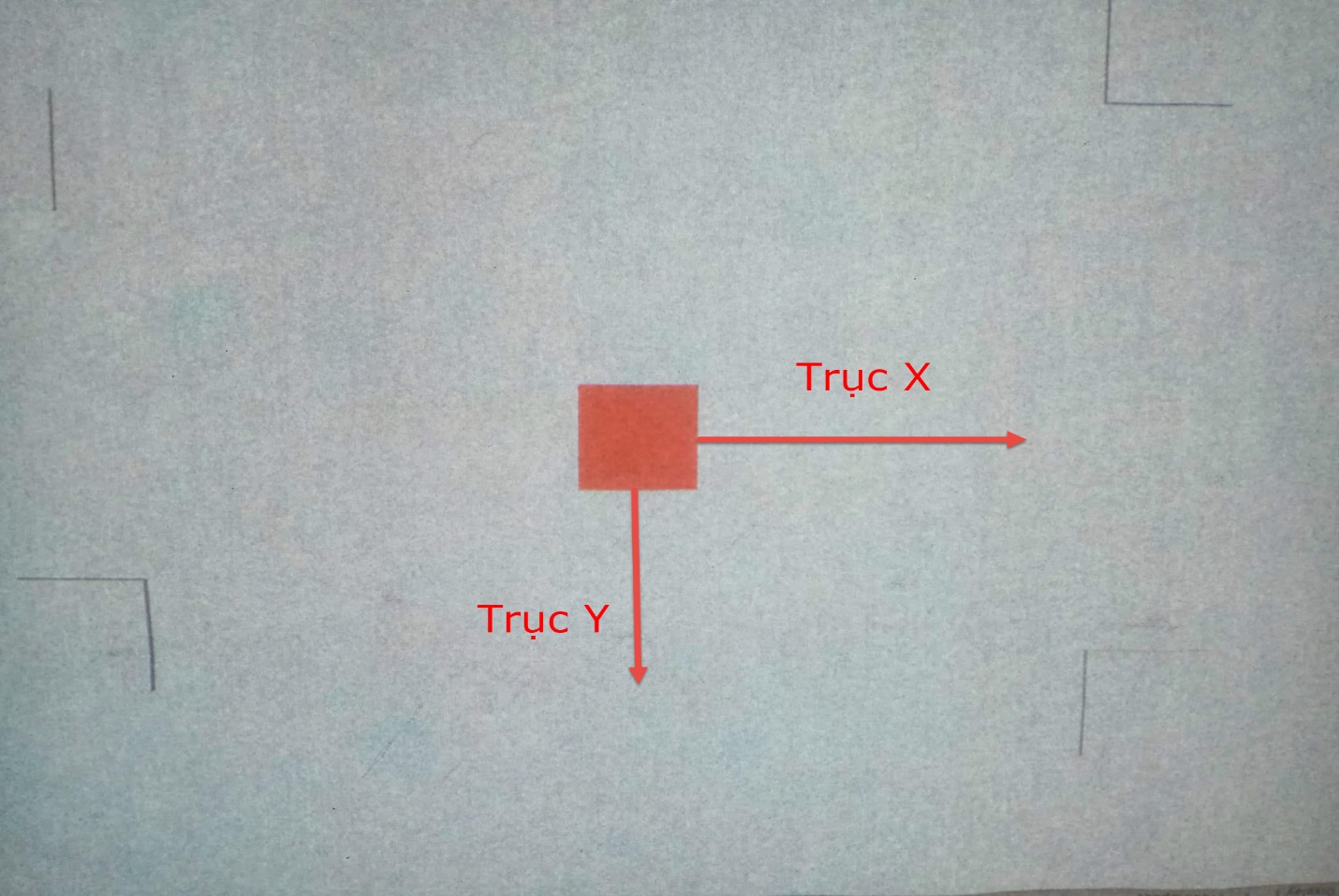


Hình 15 Xác định vùng thao tác của hệ thống

Đầu tiên, chúng em chuyển góc tọa độ ở góc trái trên bức ảnh thu được vào tâm của vùng thao tác để giống với vị trí góc tọa độ trong Unity. Chúng em đo tọa độ của góc trái trên và góc phải dưới của vùng screen (vùng thao tác) sẽ được phát ra theo tọa độ trên tấm ảnh (Hình 15). Từ đó có thể tính ra được tọa độ tâm của vùng thao tác và dùng một phép tịnh tiến tọa độ để tính tọa độ của một điểm theo gốc tọa độ mới như sau:

Gọi là tọa độ tâm của vùng thao tác ở hệ tọa độ cũ, là tọa độ của một điểm trong hệ tọa độ cũ, là tọa độ của điểm đó trong hệ tọa độ mới thì ta có:

Sau khi đã chuyển hệ tọa độ, chúng em tiến hành mapping tọa độ trên bức ảnh ở hệ trục mới, với tọa độ của màn hình trong Unity. Tư tưởng tiến hành mapping như sau: Đầu tiên chúng em tạo 1 GameObject có màu đỏ tại tâm của vùng thao tác, với kích thước 4cm x 4cm. Chúng em dùng Blob để gửi tọa độ ảnh của Object đó liên tục từ frame hình thu được qua socket vào Unity, với mọi tọa độ nhận được thì tiến hành chuyển hệ trục như công thức trên (vẫn là tọa độ ảnh).



Hình 16 Lấy mẫu Calibraiton theo 2 trục X, Y

Chúng em lấy 9 điểm tọa độ theo trục X để mapping là ; 6 điểm theo trục Y (vì tỉ lệ màn hình nên theo chiều dọc lấy được 6 điểm từ tâm màn hình) là (Hình 16). Đây là các tọa độ màn hình trong Unity, với mỗi một tọa độ màn hình sẽ tương ứng với 1 tọa độ ảnh từ Blob gửi về. Để lọc nhiễu trong quá trình lấy tọa độ, thì với mỗi tọa độ màn hình, chúng em sẽ lấy 90 tọa độ ảnh từ Blob và lấy trunsg bình của 90 giá trị đó. Sau khi đã lấy hoàn tất 15 điểm thì chúng em tiến hành mapping theo công thức sau:

Gọi là tọa độ nhận được và đã qua việc chuyển tọa độ. là tọa độ trong màn hình Unity tương ứng với tọa độ nhận được, chúng em thiết lập công thức sau:

Chúng ta cần tìm được trung bình. Với kết quả các cặp tọa độ thu được trong việc đo đạt phía trên, chúng em có thể tính được giá trị trung bình chấp nhận được của . Từ đó suy ngược được công thức tính tọa độ 1 điểm trong Unity theo tọa độ nhận được như sau:

#### 3.2.5 Kỹ thuật đồ họa trong Unity

##### Particle system

Particle system (hệ thống hạt hiệu ứng) là kĩ thuật để tạo ra cái hiệu ứng tia lửa, cháy, nổ, khói bụi, hơi nước,.. trong game. Trong game 3D phần lớn các thành phần trong cảnh, các nhân vật đều được đại diện bởi Mesh, tương tự như việc ở game 2D sử dụng sprite để tạo nên các thành phần trong khung hình. Nói cách khác các thành phần này được cấu thành bởi một vật rắn có hình dạng cho trước. Nhưng với những hiệu ứng như chất lỏng, khói, mây, lửa thì phải tiếp cận bằng cách khác, chính vì thế Particle system được tạo ra để giải quyết vấn đề này. Một vài thành phần cơ bản của Particle system:

* Duration: thời gian tính bằng giây quãng đời tồn tại của một hạt.
* Looping: có được tạo lại hay không khi hạt cuối cùng bị biến mất.
* Speed, Direction, Rotation: tốc độ, hướng di chuyển, góc quay của các hạt.
* Max particles: số hạt tối đa xuất hiện trong particle system.
* Material/Shader: dùng để thay đổi hình dáng của các hạt.
* Shape: hình dáng đường đi của các hạt tạo thành khi phát ra.

Hình ảnh trong hệ thống : …

##### Materialhttp://kipalog.kaopiz.com/uploads/0a42/6cd6/shaderintro01.png

Sơ đồ 7 Cơ chế kết suất đồ họa trong Unity

Nguốn : <https://unity3d.com/learn/tutorials/topics/graphics/gentle-introduction-shaders>

Material được sử dụng trong các thành phần kết xuất đồ họa trong Unity. Nói cách khác, nó đóng vai trò quan trọng trong việc cho thấy một Object sẽ được phô bày trên màn hình thế nào, nó như là lớp áo của Object đó. Thuộc tính mà Material thể hiện được quyết định bằng Shader. Shader là một chương trình đồ họa đặc biệt được chạy trên GPU có thể xác định bề mặt họa tiết (Texture) và thông tin ánh sáng được thể hiện thế nào khi một Object được tạo ra trên màn hình. Texture là họa tiết được lợp lên trên vật bởi Material đó. Một Shader có thể sử dụng cho nhiều Material và một Material có thể áp dụng cho nhiều 3D Object. Có thể mình họa bằng Hình 17.

##### Camera



Hình 17 Sử dụng Material để lợp màu cho Object

Nguồn : <https://docs.unity3d.com/Manual/Materials.html>

Camera là thành phần cơ bản nhất trong Unity. Điều chỉnh camera sẽ tác động đến nội dung được chiếu ra trên màn ảnh. Có thể nói đến một số thành phần cơ bản như Hình 18:



Hình 18 Thông số cơ bản của Camera

* Góc chiếu của camera (Projection): Gốm phép chiếu song song (Orthographic) và phép chiếu xa gần (Perspective). Phép chiếu song song được sử dụng trong game 2D, khi tỉ lệ các đối tượng sẽ được thể hiện không đổi ở mọi chiều sâu. Phép chiếu xa gần được sử dụng trong game 3D, mô phỏng góc nhìn theo thế giới thực, tỉ lệ các đối tượng sẽ theo luật xa gần.
* Góc mở của camera (Field Of View): thuộc tính của phép chiếu xa gần, cùng một đối tượng và cùng một vị trí, nếu góc mở càng lớn thì đối tượng càng nhỏ và ngược lại. Đây là kĩ thuật được dùng cho hiệu ứng zooming.
* Mặt phẳng xa nhất (Far) và mặt phẳng gần nhất (Near): vật chỉ được hiển thị nếu nó nằm giữa 2 mặt phẳng trên và ngược lại.

##### Colider

Trong Unity việc xử lí các tương tác vật lí sẽ giúp chương trình trở nên thực tế hơn. Trong tương tác vật lí thì xử lý va chạm là một nhu cầu cần thiết. Collider là vật mà Engine vật lý dùng để nhận ra sự va chạm. Đa số các collider có hình dạng đơn giản nhằm mục đích tính toán đơn giản và dễ dàng. Phần lớn các Object cơ bản trong Unity sẽ được gắn collider khi tạo ra (Cube, Sphere, Cylinder,…) (Hình 19). Ở hệ thống của chúng em, collider được dùng để xác định khi một Object trong game bị đè lên bởi một Object khác thông qua Raycasting (được nói ở phần sau).



Hình 19 Collider với Object Cube - Nguồn : <https://viblo.asia/p/co-ban-ve-engine-vat-ly-trong-unity-3OEqGj1PM9bL>

##### 3.2.5.5 Raycasting

Là kĩ thuật bắn một tia từ điểm *origin*, theo hướng *direction*, có chiếu dài *maxDistance* đập vào các colliders (Hình 20). Ứng dụng của nó có thể kể đến là bắt va chạm của một viên đạn với một Object khác hoặc là biết tay bạn đang chạm vào vật nào. Nó thường được sử dụng trong việc xác định đường ngắm. Ở hệ thống của chúng em, việc sử dụng Raycasting dùng để tạo một sự kiện khi tọa độ của vật chúng em cần nhận diện qua camera đè lên Object trên màn hình. Để lọc ra những vật đang quan tâm khi có Raycast chiếu trúng thì nên tạo Tag cho các vật đó để không bị nhầm lẫn với các vật khác.

* Hình 20 Minh họa Raycasting trong Unity <http://www.clonefactor.com/wordpress/unity3d-show-room/raycast-gizmos-visualizer/>

### 3.3 Các chức năng hỗ trợ

#### 3.3.1 Cho xem camera tại giao lộ [3]

#### 3.3.2

## 4. Kết luận

TÀI LIỆU THAM KHẢO

[1] E. Sharlin, B. Watson, Y. Kitamura, F. Kishino, and Y. Itoh, “On tangible user interfaces, humans and spatiality,” *Pers. Ubiquitous Comput.*, vol. 8, no. 5, pp. 338–346, Sep. 2004.

[2] H. Ishii and Hiroshi, “The tangible user interface and its evolution,” *Commun. ACM*, vol. 51, no. 6, p. 32, Jun. 2008.

[3] E. Hornecker and J. Buur, “Getting a grip on tangible interaction,” in *Proceedings of the SIGCHI conference on Human Factors in computing systems - CHI ’06*, 2006, p. 437.