TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN CÔNG NGHỆ PHẦN MỀM

LÊ ĐÌNH HÂN – TRẦN PHƯỚC HIỀN

ỨNG DỤNG ABC

KHOÁ LUẬN TỐT NGHIỆP CỬ NHÂN CNTT

x

TP.HCM, 2018

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN CÔNG NGHỆ PHẦN MỀM

LÊ ĐÌNH HÂN 1412153

TRẦN PHƯỚC HIỀN 1412171

ỨNG DỤNG ABC

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN TIN HỌC

GIÁO VIÊN HƯỚNG DẪN

ThS. TRẦN QUANG TÁNH – ThS. ĐẶNG BÌNH PHƯƠNG

NIÊN KHÓA 2014 – 2018

|  |
| --- |
| **NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  Khóa luận đáp ứng yêu cầu của LV cử nhân tin học.  TP.HCM, ngày …… tháng …… năm 2018  Giáo viên hướng dẫn |
| **NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN**  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  Khóa luận đáp ứng yêu cầu của LV cử nhân tin học.  TP.HCM, ngày …… tháng …… năm 2018  Giáo viên phản biện |

LỜI CÁM ƠN

Chúng em xin chân thành cảm ơn Khoa Công Nghệ Thông Tin, trường Đại Học Khoa Học Tự Nhiên, Tp.HCM đã tạo điều kiện tốt cho chúng em thực hiện đề tài này.

Chúng em xin chân thành cảm ơn Thầy Trần Quang Tánh và Thầy Đặng Bình Phương đã tận tình hướng dẫn, chỉ bảo chúng em trong suốt thời gian thực hiện đề tài.

Chúng em cũng xin gửi lời cảm ơn sâu sắc đến quý Thầy Cô trong Khoa đã tận tình giảng dạy, trang bị cho chúng em những kiến thức quý báu trong những năm học vừa qua.

Chúng em xin gửi lòng biết ơn sâu sắc đến Ba, Mẹ, các anh chị và bạn bè đã ủng hộ, giúp đỡ và động viên chúng em trong những lúc khó khăn cũng như trong suốt thời gian học tập và nghiên cứu.

Mặc dù chúng em đã cố gắng hoàn thành luận văn trong phạm vi và khả năng cho phép, nhưng chắc chắn sẽ không tránh khỏi những thiếu sót, kính mong sự cảm thông và tận tình chỉ bảo của quý Thầy Cô và các bạn.

Nhóm thực hiện

ĐỀ CƯƠNG CHI TIẾT

|  |
| --- |
| **Tên Đề Tài** : Ứng dụng abc |
| **Giáo viên hướng dẫn**: ThS. Trần Quang Tánh – ThS. Đặng Bình Phương |
| **Thời gian thực hiện** : từ ngày aa/bb/2018 đến ngày aa/bb/2018 |
| **Sinh viên thực hiện :**  Lê Đình Hân (1412153) – Trần Phước Hiền (1412171) |
| **Loại đề tài :** Tìm hiểu công nghệ và xây dựng ứng dụng |
| **Nội Dung Đề Tài** : Mục tiêu của đề tài được xác định là *???*  Nội dung chi tiết của luận văn bao gồm : |

|  |  |
| --- | --- |
| **Kế Hoạch Thực Hiện:**   * aa/aa/2018-aa/aa/2018 | |
| **Xác nhận của GVHD** | **Ngày tháng 7 năm 2018**  **Nhóm SV Thực hiện**  Lê Đình Hân – Trần Phước Hiền |

MỤC LỤC

[LỜI CÁM ƠN iii](#_Toc515781194)

[ĐỀ CƯƠNG CHI TIẾT iv](#_Toc515781195)

[MỤC LỤC vi](#_Toc515781196)

[DANH MỤC CÁC HÌNH ix](#_Toc515781197)

[DANH MỤC MÃ xi](#_Toc515781198)

[DANH MỤC CÁC SƠ ĐỒ xii](#_Toc515781199)

[TÓM TẮT KHÓA LUẬN xiii](#_Toc515781200)

[Chương 1 Mở đầu 1](#_Toc515781201)

[1.1. Giới thiệu chung 1](#_Toc515781202)

[1.1.1. Tangible và môi trường tương tác Tangible 1](#_Toc515781203)

[1.1.2. Các kỹ thuật tương tác trong môi trường hỗ trợ tương tác Tangible 1](#_Toc515781204)

[1.2. Lý do thực hiện đề tài 1](#_Toc515781205)

[1.3. Mục tiêu đề tài 1](#_Toc515781206)

[1.4. Nội dung luận văn 2](#_Toc515781207)

[Chương 2 Các phương pháp nhận dạng vật thể 3](#_Toc515781208)

[2.1. Tổng quan về nhận dạng 3](#_Toc515781209)

[2.2. Nhận dạng vật bằng màu sắc 3](#_Toc515781210)

[2.3. Áp dụng Vuforia trong việc nhận dạng 6](#_Toc515781211)

[2.3.1. Giới thiệu 6](#_Toc515781212)

[2.3.2. Các vấn đề kỹ thuật khi sử dụng Vuforia 6](#_Toc515781213)

[2.4. Áp dụng Kinect trong việc nhận dạng 12](#_Toc515781214)

[2.4.1. Giới thiệu Kinect 12](#_Toc515781215)

[2.4.2. Cấu tạo Kinect v2 14](#_Toc515781216)

[2.4.3. Các vấn đề kỹ thuật khi sử dụng kinect 16](#_Toc515781217)

[2.5. Kết luận 16](#_Toc515781218)

[Chương 3 Các vấn đề kỹ thuật trong Unity 17](#_Toc515781219)

[3.1. Mở đầu 17](#_Toc515781220)

[3.2. Các vấn đề kỹ thuật và giải pháp trong Unity 18](#_Toc515781221)

[3.2.1. Camera 18](#_Toc515781222)

[3.2.2. Material 19](#_Toc515781223)

[3.2.3. Colider 21](#_Toc515781224)

[3.2.4. Particle system 22](#_Toc515781225)

[3.2.5. Raycasting 23](#_Toc515781226)

[3.2.6. Lighting 23](#_Toc515781227)

[3.2.7. Position 26](#_Toc515781228)

[3.3. Kết luận 27](#_Toc515781229)

[Chương 4 Hệ thống A 28](#_Toc515781230)

[4.1. Giới thiệu tổng quan 28](#_Toc515781231)

[4.2. Cấu hình hệ thống 29](#_Toc515781232)

[4.3. Kịch bản sử dụng hệ thống 31](#_Toc515781233)

[4.4. Một số vấn đề gặp phải và kỹ thuật áp dụng 31](#_Toc515781234)

[4.4.1. Cách bố trí máy chiếu 31](#_Toc515781235)

[4.4.2. Kỹ thuật ánh xạ tọa độ giữa các không gian tọa độ 34](#_Toc515781236)

[4.4.3. Kiến trúc hệ thống sự kiện liên hoàn 37](#_Toc515781237)

[4.4.4. Giao tiếp giữa các thiết bị tương tác 42](#_Toc515781238)

[4.4.5. Kiến trúc MVP trên các thiết bị di động 44](#_Toc515781239)

[4.5. Các chức năng của hệ thống 47](#_Toc515781240)

[4.5.1. Các cách thức tương tác 47](#_Toc515781241)

[4.5.2. Màn hình giao diện chính 47](#_Toc515781242)

[4.6. Kết luận 47](#_Toc515781243)

[TÀI LIỆU THAM KHẢO 48](#_Toc515781244)

DANH MỤC CÁC HÌNH

[Hình 2‑1 Kết quả các bước trong quy trình nhận dạng vật thể 4](#_Toc515781245)

[Hình 2‑2 Ví dụ về toán tử erosion 5](#_Toc515781246)

[Hình 2‑3 Đăng kí tài khoản Vuforia 6](#_Toc515781247)

[Hình 2‑4 Cho phép sử dụng Vuforia trong project 7](#_Toc515781248)

[Hình 2‑5 Giao diện lấy App Key 8](#_Toc515781249)

[Hình 2‑6 Gắn App Key vào project 8](#_Toc515781250)

[Hình 2‑7 Tạo Database hình ảnh nhận diện 9](#_Toc515781251)

[Hình 2‑8 Tải Database về project 9](#_Toc515781252)

[Hình 2‑9 Chọn Target theo hình ảnh từ Database 9](#_Toc515781253)

[Hình 2‑10 Dựa vào độ tương phản 10](#_Toc515781254)

[Hình 2‑11 Độ tương phản ảnh Grayscale tương ứng với histogram 10](#_Toc515781255)

[Hình 2‑12 Cấu tạo Kinect v2 13](#_Toc515781256)

[Hình 2‑13 Image stream Kinectv2 14](#_Toc515781257)

[Hình 2‑14 Depth stream kinectv2 15](#_Toc515781258)

[Hình 2‑15 Body stream Kinectv2 15](#_Toc515781259)

[Hình 3‑1 Giao diện Unity Engine 17](#_Toc515781260)

[Hình 3‑2 Thông số cơ bản của Camera 19](#_Toc515781261)

[Hình 3‑3 Sử dụng Material để lợp màu cho Object Nguồn : https://docs.unity3d.com/Manual/Materials.html 21](#_Toc515781262)

[Hình 3‑4 Colider với Object Cube 22](#_Toc515781263)

[Hình 3‑5 Minh họa Raycasting trong Unity 23](#_Toc515781264)

[Hình 3‑6 Các thành phần cơ bản của Light Component 24](#_Toc515781265)

[Hình 3‑7 Point Light 25](#_Toc515781266)

[Hình 3‑8 Spot Light 25](#_Toc515781267)

[Hình 3‑9 Directional Light 25](#_Toc515781268)

[Hình 3‑10 Area Light 25](#_Toc515781269)

[Hình 4‑1 Cấu hình thiết lập của hệ thống 29](#_Toc515781270)

[Hình 4‑2 Hình ảnh thực tế của hệ thống 30](#_Toc515781271)

[Hình 4‑3 Cách bố trí máy chiếu trên cao 31](#_Toc515781272)

[Hình 4‑4 Cách bố trí máy chiếu dưới đất theo phương vuông góc 32](#_Toc515781273)

[Hình 4‑5 Cách bố trí máy chiếu dưới đất kém theo gương 33](#_Toc515781274)

[Hình 4‑6 Xác định vùng thao tác của hệ thống 35](#_Toc515781275)

[Hình 4‑7 Lấy mẫu Calibration theo 2 trục X, Y 36](#_Toc515781276)

[Hình 4‑8 Sơ đồ mô hình MVC 45](#_Toc515781277)

[Hình 4‑9 Sơ đồ mô hình MVP 45](#_Toc515781278)

DANH MỤC MÃ

[Mã 1 Minh họa một Comparator 35](#_Toc515741873)

[Mã 2 Minh họa một Action 36](#_Toc515741874)

[Mã 3 Minh họa một Operator 37](#_Toc515741875)

DANH MỤC CÁC SƠ ĐỒ

[Sơ đồ 3‑1 Cơ chế kết xuất đồ họa trong Unity 6](#_Toc515732772)

[Sơ đồ 4‑1 Hệ thống sự kiện 21](#_Toc515732773)

TÓM TẮT KHÓA LUẬN

Ngày nay,

# Mở đầu

* Nội dung Chương 1 trình bày tổng quan về đề tài, lý do thực hiện đề tài, từ đó nêu rõ mục tiêu của khóa luận. Nội dung tóm tắt của từng chương trong khóa luận được trình bày ở cuối chương này.

## Giới thiệu chung

### Tangible và môi trường tương tác Tangible

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]

### Các kỹ thuật tương tác trong môi trường hỗ trợ tương tác Tangible

* Kỹ thuật A

Kỹ thuật

* Kỹ thuật B

Kỹ thuật B

* Kết luận

Tương … . Chúng em sẽ trình bày rõ hơn về lý do chúng em thực hiện đề tài ở phần tiếp theo.

## Lý do thực hiện đề tài

## Mục tiêu đề tài

Đề tài chúng

## Nội dung luận văn

Luận văn bao gồm X chương, sau đây là nội dung chính của từng chương.

**Chương 1:**

**Chương 2:**

**Chương 3**

**Chương 7:** trình bày các kết quả đã đạt được trong quá trình thực hiện đề tài và hướng phát triển trong tương lai.

# Các phương pháp nhận dạng vật thể

* Nội dung Chương 2 đề xuất, phân tích và thực nghiệm các giải pháp để nhận diện vật thể. Từ đó nêu lên ưu điểm, hạn chế của từng phương pháp và áp dụng các phương pháp này để xây dựng một môi trường tương tác hỗ trợ Tangible

## Tổng quan về vấn đề nhận dạng

* Đặt vấn đề

Việc nhận dạng chính xác vật thể đóng vai trò quan trong việc xây dựng môi trường tương tác có hỗ trợ Tangible. Nhận dạng là công cụ gián tiếp để các hành vi, thao tác của người dùng giao tiếp được vời hệ thống. Làm sao để xác định sự xuất hiện của một vật ? Làm sao để truy vết được sự thay đổi vị trí của vật đó ?

* Giải pháp

Để trả lời cho các câu hỏi trên, chúng em xin đề xuất ra các phương pháp tiếp cận dưới đây, từ đó tiến thành thực nghiệm, phân tích, đánh giá và đưa ra kết luận phương pháp phù hợp nhất:

Nhận dạng bằng màu đơn sắc (Chương 2.2)

Nhận dạng bằng họa tiết ảnh áp dụng Vuforia (Chương 2.3)

Nhận dạng bằng ảnh hồng ngoại kết hợp thông tin độ sâu (Chương 2.4)

## Nhận dạng vật bằng ảnh màu

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 2‑1 Quy trình nhận dạng vật thể bằng xử lý ảnh cơ bản

Sơ đồ 2‑1 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 2‑1. B. Sau đó đem chồng lên 1 lớp ảnh Mask (Hình 2‑1. A) đã lọc theo ngưỡng màu từ ảnh gốc. Kết quả thu được ta sẽ thu được một ảnh trắng đen như Hình 2‑1. 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 2‑1 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.



Hình 2‑2 Ví dụ về toán tử erosion

Một ví dụ trực quan hơn như Hình 2‑2, 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.

* Toán tử dilation

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.

## Áp dụng Vuforia trong việc nhận dạng

### Giới thiệu

### Các vấn đề kỹ thuật khi sử dụng Vuforia

#### Tích hợp 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 :

1. Truy cập vào <https://developer.vuforia.com/vui/auth/register> để đăng kí tài khoản developer (Hình 2‑3). 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 2‑3 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>

1. 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 2‑4)

Hình 2‑4 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 2‑5)

Ta copy phần App Key, ở màn hình Unity, ta chọn Window => Vuforia Configuration, sau đó Key vào ô App License Key (Hình 2‑6).



Hình 2‑5 Giao diện lấy App Key



Hình 2‑6 Gắn App Key vào project

1. Chuẩn bị database cho việc nhận dạng

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.

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 2‑7). 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 2‑8).



Hình 2‑7 Tạo Database hình ảnh nhận diện



Hình 2‑8 Tải Database về project

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 2‑9 Chọn Target theo hình ảnh từ Database

#### Tiêu chí đánh giá ảnh của Vuforia và các yếu tố ảnh hưởng



Hình 2‑10 Dựa vào độ tương phản

Nguồn : <https://www.edrawsoft.com/umldiagramtemplate.php>



Hình 2‑11 Độ tương phản ảnh Grayscale tương ứng với histogram

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 (Hình 2‑10), 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 (Hình 2‑11)

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 2‑1 và khả năng ánh xạ tọa độ thu được của vật trong thế giới ảo với vật trong thế giới thật khá phức tạp.

## Áp dụng Kinect trong việc nhận dạng

### Giới thiệu Kinect

Kinect là thiết bị cảm biến thu nhận chuyển động được Microsoft phát triển dành riêng cho máy chơi game Xbox360 và về sau là nhằm phục vụ máy vi tính sử dụng hệ điều hành Windows. Quá trình phát triển của Kinect có thể tóm tắt như sau: Vào 30/5/2007, trong hội nghị D5, chủ tịch Microsoft là Bill Gates đã đề xuất ra một phương thức điều khiển game từ người chơi mới dựa vào thu nhận dữ liệu từ camera, tương tác trực tiếp bằng hành động cử chỉ của người chơi và vật thể thực tế. Ngày 7/4/2008, MTV News Reports đưa tin Microsoft đang tích cực phát triển một thiết bị điều khiển bằng chuyển động để cạnh tranh với Nintendo’s Wii. 1/6/2009, Microsoft công bố dự án Natal tại hội nghị E3, đồng thời đưa ra demo cho việc sử dụng thiết bị với công nghệ điều khiển bằng hành động. Natal có thể nhận diện được đồng thời 4 người với 48 khớp xương trên cơ thể. Đến ngày 13/6/2010, tại E3, Microsoft chính thức đổi tên dự án Natal thành Kinect kết hợp giữa 2 từ kinectic và connect. Ngày 20/7/2010, Microsoft thông báo chính thức thời gian phát hành và giá bán của Kinect, nhưng mới chỉ dành cho phiên bản Xbox. Ngày 2/1/2012, Microsoft công bố phiên bản Kinect dành cho windows kèm theo phiên bản thương mại chính thức Kinect SDK. Đến tháng 10/2014, phiên bản thứ 2 của Kinect được bày bán với những cải tiến:

* Độ phân giải Full HD 1080.
* Tích hợp bộ cảm biến 3D, tăng khả năng nắm bắt chi tiết.
* Cảm biến hồng ngoại để nhận biết được đối tượng trong môi trường thiếu sáng, với phạm vị hoạt động rộng hơn.
* Độ sâu trung thực (3x).
* Tăng điểm khung xương lên 25.
* Cải thiện tầm nhìn (0.5m đến 4.5m).
* Đa ứng dụng cùng xử dụng 1 Kinect.
* Hỗ trợ theo dõi các ngón tay, cử chỉ nắm mở bàn tay.
* Có khả năng đo được nhịp tim của người dùng.
* So sánh Kinect v2 và Kinect v1:

|  |  |  |
| --- | --- | --- |
| **Đặc điểm** | **Kinect v1** | **Kinect v2** |
| Màu sắc | 640×480 @ 30fps | 1920×1080 @ 30fps |
| Độ sâu | 320×240 @ 30fps | 512×424 @ 30fps |
| Cảm biến | Structured Light  (PrimeSense Light Coding) | Time of Flight (ToF) |
| Tầm nhìn | 0.8~4.0 m | 0.5~4.5 m |
| Góc nhìn (ngang/dọc) | 570/430 | 700/600 |
| Mic | Có | Có |
| Số người nhận được | 6 | 6 |
| Số khớp xương | 20 | 25 |
| Trạng thái tay | Open/Close | Open/Close/Lasso |
| Cử chỉ | Không hỗ trợ | Có |



Hình 2‑12 Cấu tạo Kinect v2

Nguồn : <https://www.slideshare.net/MarcoDAlessandro/nui-e-biometrics-in-windows-10>

### Cấu tạo Kinect v2

Kinect v2 có kích thước 14.8” x 5.9” x 4.8”, có một camera RGB, một cảm biến độ sâu và dãy microphone. Nó cho phép thu được đầy đủ chuyển động cơ thể dưới dạng 3D, nhận dạng khuôn mặt và giọng nói. Camera RGB cho phép nhận diện tối đa 6 người cùng lúc, theo dõi chuyển động tối đa 2 người. Có cấu tạo được thể hiện như Hình 2‑12 :

* IR Emitters: thiết bị dùng để phát tia hồng ngoại.
* RGB Camera: camera dùng để thu hình ảnh màu bên ngoài. Camera này hỗ trợ nhiều định dạng ảnh và độ phân giải khác nhau
* IR Camera: có nhiệm vụ bắt tia hồng ngoại được chiếu từ IR Emitters phản xạ từ các vật thể và dùng thông tin đó để tính toán khoảng cách từ camera đến các vật thể được chiếu đến.
* Multi-Array Mic: là một dãy gồm 4 microphone có nhiệm vụ thu lại âm thanh và xác định hướng phát ra âm thanh. Dãy microphone này được dùng trong các ứng dụng cần nhận diện giọng nói.

Các định dạng dữ liệu stream được Kinect v2 hỗ trợ :

* Image Stream: trả về dữ liệu ảnh màu thu được từ RGB camera, hỗ trợ định dạng hình ảnh RGB\_1920x1080(định dạng RGB, độ phân giải 1920x1080, tốc độ khung hình tối đa 30 fps, cho phép khả năng tăng độ sáng, phơi sáng tốt hơn).



Hình 2‑13 Image stream Kinectv2

Nguồn : <https://www.mindtreatstudios.com/the-art-of-speech/>

* Depth Stream: trả về dữ liệu độ sâu của khung nhìn và dữ liệu phân đoạn cơ thể bằng cách sử dụng 3D Depth Sensor. Với body tracking, có thể lấy được hình ảnh của 6 người tại một khung hình trong khoảng từ 0.5ms đến 4.5m. Depth Stream hỗ trợ định dạng DEPTH\_512x424(độ phân giải 512x424, tốc độ khung hình 30fps).



Hình 2‑14 Depth stream kinectv2

Nguồn : <https://brekel.com/kinect-2-details-specifications-observations/>

* Body Stream: trả về dữ liệu độ sâu với mốc thời gian, bao gồm vị trí tổng thể của cơ thể với 25 khớp xương



Hình 2‑15 Body stream Kinectv2

Nguồn : <https://pterneas.com/2014/03/13/kinect-for-windows-version-2-body-tracking/>

### Các vấn đề kỹ thuật khi sử dụng Kinect

#### Lấy các kênh ảnh từ kinect

* Đặt vấn đề

Một lợi thế rất lớn khi sử dụng Kinect là Kinect cung cấp cho chúng ta nhiều kênh thông tin khác nhau để phục vụ cho việc nhận dạng. Ở đây, chúng em xác định sử dụng 2 kênh hình ảnh là ảnh hồng ngoại và ảnh độ sâu. Vậy làm sao có thể giao tiếp với Kinect để lấy thông tin từ các kênh ảnh này ?

* Giải pháp

Để lấy thông tin ảnh hồng ngoại từ Kinect, ta chỉ cần khởi tạo đối tượng *KinectSensor* và gọi API InfraredFrameSource. Trong luận văn nảy, chúng em sử dụng Kinectv2 nên ảnh trả về có kích thước là 512x424. Tuy nhiên để tăng tốc xử lý, ta có thể giới hạn lại kích thước ảnh trả về để hình ảnh trả về phù hợp với không gian cần nhận dạng.

## Kết luận

So sánh giữa 3 phương pháp -> đưa ra kết luận

# Các vấn đề kỹ thuật trong Unity

* Chương 3 sẽ trình bày các vấn đề về kỹ thuật trong Unity, bao gồm: nhu cầu thực tế phát sinh; giải pháp đưa ra; cách thức tiến hành, cài đặt trong Unity và các thông số kỹ thuật trong Unity

## Các vấn đề kỹ thuật và giải pháp trong Unity

### Camera

* Vấn đề

Để có thể linh hoạt trong việc trình chiếu cảnh vật, đối tượng, ta cần phải điều chỉnh hướng góc nhìn, tầm nhìn hoặc thay đổi cả vị trí của góc nhìn như góc nhìn thứ nhất, góc nhìn thứ ba. Ở mỗi góc nhìn khác nhau, người dùng sẽ có những trải nghiệm khác nhau. Vậy trong Unity có công cụ nào để tùy chỉnh góc nhìn hay không ?

* Giải pháp

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 3‑1 :

* 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.



Hình 3‑2 Thông số cơ bản của Camera

## Mở đầu



Hình 3‑1 Giao diện Unity Engine

Unity là một nền tảng hỗ phát triển game đa nền tảng được phát triển bởi Unity Technologies [2] Lần đầu tiên Unity được công bố chạy trên hệ điều hành OS X, tại Apple's Worldwide Developers Conference vào năm 2005, đến nay đã mở rộng 27 nền tảng. Unity cung cấp một công cụ soạn thảo bằng giao diện người dùng dễ sử dụng. Đặt biệt Unity hỗ trợ rất tốt các công việc liên quan đến đồ họa 3D, người dùng có thể dễ dàng kéo và thả các đối tượng, tùy chỉnh các thông số của mỗi đối tượng qua hệ thống menu (Hình 3‑1 Giao diện Unity Engine). Bên cạnh dùng giao diện, người dùng Unity còn có thể viết kịch bản bằng các ngôn ngữ JavaScript, C # hoặc Boo. Hơn nữa, Unity có một cộng đồng rất lớn trên Unity Store, nơi cung cấp rất nhiều asset, resource có sẵn từ miễn phí đến có phí.

### Material

* Vấn đề

Với một Object được tạo ra thì thành phần cơ bản của nó là Mesh giúp nó xác định được hình dạng, để hỗ trợ cho việc thể hiện thêm thông tin màu sắc thì theo cơ chế kết xuất đồ họa trong Unity (Sơ đổ 7) thì ta cần sử dụng một công cụ giúp ta trang trí vật đó, nó là Material.

* Giải pháp



Sơ đồ 3‑1 Cơ chế kết xuất đồ họa trong Unity

Nguồn : <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.



Hình 3‑3 Sử dụng Material để lợp màu cho Object

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

### Colider

* Vấn đề

Để mối quan hệ giữa các đối tượng trong ứng dụng trở nên thực hơn, ta phát sinh ra nhu cầu áp dụng tính chất vật lý lên các đổi tượng như rơi, va chạm, lục kéo ma sát,.. Vậy để áp dụng các tính chất vậy lý này, Unity có công cụ nào để hỗ trợ hay không ?

* Giải pháp

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 3‑4 Colider với Object Cube

Nguồn : <https://viblo.asia/p/co-ban-ve-engine-vat-ly-trong-unity-3OEqGj1PM9bL>

### Particle system

* Vấn đề

Để nâng cao trải nghiệm người dùng khi thao tác trong môi trường tương tác có hỗ trợ Tangible, đôi khi ta cần tạo ra các hiệu ứng đẹp mắt như hào quang, khói, lửa,.. Các hiệu ứng này giúp thu hút sự chú ý của người dùng cũng như là một dấu hiệu để gợi ý cho người dùng trong các hệ thống hoạt động có kịch bản. Vậy trong Unity có công cụ nào để hỗ trợ tạo nên các hiệu ứng này hay không.

* Giải pháp

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. Hệ thống của chúng em cũng áp dụng particle system để tạo ra các hiệu ứng màu sắc cho việc chọn một Object để gây sự chú ý của người dùng trong quá trình sử dụng. 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.

### Raycasting

* Vấn đề

Trong quá trình phát triển hệ thống, chúng em phát sinh nhu cầu làm thế nào để tìm được hình chiếu của một điểm lên một mặt phẳng cho trước, nhu cầu này gần giống với hành vi teleport trong game hoặc dùng tay chỉ vào một vật để chọn vật đó. Vậy kỹ thuật này được làm như thế nào trong Unity

* Giải pháp

Trong Unity, raycasting là kĩ thuật bắn một tia từ điểm *origin*, theo hướng *direction*, có chiều dài *maxDistance* đập vào các colliders (Hình 3‑3). 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 3‑5 Minh họa Raycasting trong Unity

Nguồn : <http://www.clonefactor.com/wordpress/unity3d-show-room/raycast-gizmos-visualizer/>

### Lighting

* Đặt vấn đề

Trong thế giới thật, ánh sáng là một trong các yếu tố quan trọng nhất để tạo nên hình khối, độ chân thật các vật thể. Một mô hình 2D nếu kết hợp với việc chiếu sáng đúng góc nhìn và cường độ ánh sáng thích hợp sẽ làm cho người nhìn cảm thấy vật thật hơn. Vậy làm sao để điều chỉnh, thiết lập ánh sáng một cách chân thật nhất trong môi trường Unity ?

* Giải pháp

Ánh sáng là một thành phần thiết yếu trong mọi cảnh, nếu chỉ dùng Mesh và Texture dùng để định hình các thành phần thì vẫn không thể mang lại hiệu quả đồ họa như mong muốn. Chính vì thế ánh sáng được dùng để xác định màu sắc của môi trường 3D, nhờ có ánh sáng ta không những phân biệt được màu sắc bề mặt vật mà còn thấy được vật độ phản xạ ánh sáng của vật (độ bóng-nhám của bề mặt). Một cảnh trong Unity có thể kết hợp nhiều nguồn sáng để tạo ra hiệu ứng đặc biệt về ánh sáng và màu sắc. Các thành phần cơ bản của ánh sáng như Hình 3‑5



Hình 3‑6 Các thành phần cơ bản của Light Component

* Type : loại ánh sáng trong Unity, có 4 loại : Directional mô phỏng ánh sáng song song từ mặt trời (Hình 3‑7.C), Point mô phỏng ánh sáng từ bóng đèn, tâm nguồn sáng ở một điểm và phát tia sáng ra mọi hướng (Hình 3‑7.A), Spot mô phỏng ánh sáng từ đèn pin, từ tâm nguồn sáng sẽ phát ra chùm sáng phân kì theo hình nón (Hình 3‑7.B), Area mô phỏng ánh sáng phát ra từ bảng hiệu, ánh sáng phát ra từ mọi điểm trên 1 mặt của hình chữ nhật (Hình 3‑7.D)



A.Point Light

B.Spot Light

C.Directional Light

D.Area Light

Hình 3‑7 Các loại Light trong Unity

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

* Range : xác định ánh sáng sẽ được chiếu bao xa.
* Spot Angle : xác định độ mở của chóp nón trong Spot Light
* Color : màu ánh sáng phát ra.
* Shadow Type : cường độ của bóng khi vật bị ánh sáng chiếu vào.

Trong hệ thống chúng em sử dụng Directional Light vì mọi thao tác sẽ được thực hiện trên mặt phẳng nên tại mọi điểm trên mặt phẳng đó cần có góc chiếu sáng, độ đổ bóng, cường độ ánh sáng nhận được là như nhau.

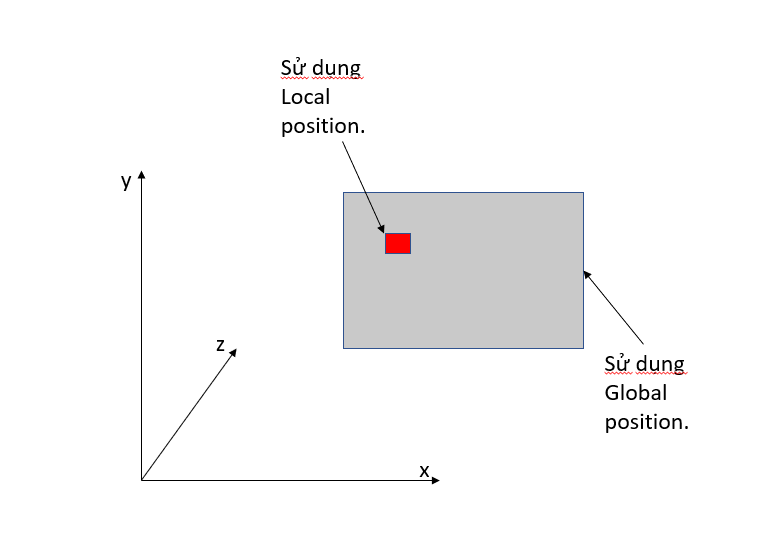
### Position

* Vấn đề

Hệ thống của chúng em đang tồn tại 2 hệ tọa độ khác nhau: một hệ tọa độ từ camera thu được, một hệ tọa độ trong môi trường Unity. Vậy làm sao để có thể ánh xạ qua lại một tọa độ giữa hai hệ trục tọa độ đó?

* Giải pháp

Khi tiến hành mapping tọa độ từ ảnh sang tọa độ Unity thì việc hiểu rõ các loại tọa độ trong Unity là rất quan trọng. Vì nó giúp chúng em dễ dàng hơn trong việc mapping tọa độ chính xác hơn. Để xác định vị trí của một Object trong Unity thì có 2 loại vị trí : position (global) và local position (local). Local position là vị trí của Object tương đối với vị trí của Parent Object chứa nó. Global position là vị trí của Object tính theo gốc tọa độ của World Space. Với những thành phần tách biệt nhau thì nên sử dụng Global position, tuy nhiên với việc ánh xạ tọa độ cho phù hợp với màn hình thì ở đây chúng em sử dụng Local position để dễ tính toán. Chúng em sử dụng một Plane Object để tượng trưng cho màn hình, sau đó đặt các Object con cần thao tác là con của Plane đó. Khi sử dụng Local position thì việc tính tọa độ sẽ dựa vào tâm của Plane giống như hệ tọa độ khi lấy từ camera, chính vì thế sẽ dễ dàng hơn cho việc ánh xạ tọa độ (Hình 3‑8).



Hình 3‑8 Mô phỏng cách sử dụng position trong hệ thống

## Kết luận

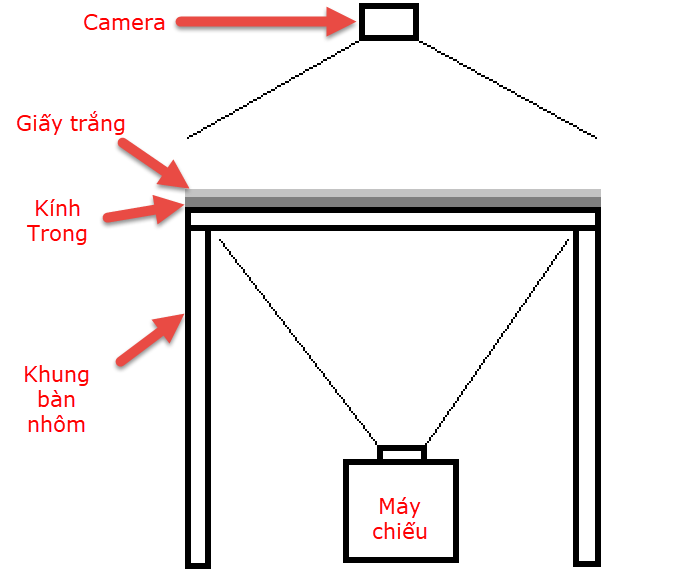
# Hệ thống A

* Chương 4 này sẽ giới thiệu các chức năng của hệ thống A

## Giới thiệu tổng quan

Hệ thống

## Cấu hình hệ thống



Hình 4‑1 Cấu hình thiết lập của hệ thống

Hình 4‑1 minh họa cấu hình tổng thể của hệ thống. Các thành phần chính của hệ thống này 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
* Một 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

Sau khi xây dựng theo các thông số trên, ta sẽ có được hệ thống thực tế như Hình 4‑2



Hình 4‑2 Hình ảnh thực tế của hệ thống

-

## Kịch bản sử dụng hệ thống

Kịch bản sử dụng

## Một số vấn đề gặp phải và kỹ thuật áp dụng

### Cách bố trí máy chiếu

* Vấn đề

Bài toán cần giải quyết là lắp đặt ví trí máy chiếu làm sao cho thích hợp để thỏa mãn 2 tiêu chí:

Chi phí lắp đặt rẻ, dễ và nhanh chóng lắp đặt

Nâng cao trải nghiệm người dùng, dễ sử dụng và đẹp mắt

* Giải pháp

Qua phân tích và thử nghiệm thực tế, chúng em đề xuất 3 cách bố trí máy chiếu khả thi nhất

Cách 1: Đặt máy chiếu trên cao, chiếu từ trên xuống

Cách 2: Đặt máy chiếu dưới mặt đất, phương vuông góc với mặt đất, chiếu từ dưới lên

Cách 3: Đặt máy chiếu dưới đất, áp dụng phản xạ gương để tạo góc chiếu 90o so với mặt đất

#### Đặt máy chiếu trên cao

**Máy chiếu**

50cm

130cm

80cm

Hình 4‑3 Cách bố trí máy chiếu trên cao

Thông số lắp đặt cụ thể được mô tả ở Hình 4‑3.Với cách lắp đặt này, ta tận dụng được giá treo hiện tại của Camera. Hình ảnh chiếu ra rất rõ nét, chân thật và sống động, mang đến cho người dùng trải nghiệm rất tốt

Tuy nhiên, nhược điểm của cách lắp đặt này là ta cần gia cố giá đỡ thêm chắc chắn vì máy chiếu khá nặng. Hơn thế nữa, việc lắp đặt khá khó khăn do phải lắp đặt cố định để máy chiếu không rung lắc trong quá trình sử dụng.Việc lắp đặt cứng này gây khó khăn nếu về sau có nhu cầu thay đổi, nâng cấp về phần cứng.

Ngoài ra, việc chiếu ánh sáng từ trên cao xuống sẽ gây nên vấn đề là gây đổ bóng lên mặt bàn khi người dùng thao tác dưới máy chiếu, làm cho thông tin chiếu ra bị bóp méo, sai lệch với kích thước ban đầu

#### Đặt máy chiếu dưới đất theo phương vuông góc mặt đất

Camera

103cm

180cm

80cm

75cm

**Máy chiếu**

Hình 4‑4 Cách bố trí máy chiếu dưới đất theo phương vuông góc

Theo cách bố trí ở Hình 4‑4, ta thấy rõ ràng cách lắp đặt này đã khắc phục được nhược điểm gây độ bóng của cách lắp đặt trước. Khung giá đỡ cũng không cần phải gia cố thêm, giúp tiết kiệm chi phí. Do mặt bàn trong trường hợp này là kính trong suốt có phủ giấy mỏng, nên hình cảnh chiếu ra từ máy chiếu tuy không sặc sỡ, thu hút như cách lắp đặt trước, nhưng vẫn hiển thị rõ thông tin ở mức chấp nhận được. Nếu thay đổi chất liệu kính trong suốt và giấy trắng thảnh kính đục để trình chiếu thì chất lượng ảnh sẽ tốt hơn nhiều. Tuy nhiên, trong giới hạn luận văn này, chúng em chỉ tiến hành thí nghiệm trên kính trong suốt và giấy A4

Tuy nhiên, cách bố trí này có khuyết điểm là việc giữ máy chiếu ở góc 90o gây ra một số khó khăn như : mặt tản nhiệt của máy nằm ở phần sau, nếu trình chiếu trong thời gian dài sẽ làm nóng máy do mặt tản nhiệt áp xuống đất, cần làm thêm một giá đỡ để giữ máy chiếu cố định. Ngoài ra vùng chiếu ra của máy chiếu nhỏ nên nếu muốn hứng được ảnh lớn hơn, ta phải nâng cao khoảng cách màn hứng và máy chiếu. Việc này khiến tổng chiều cao của cả ứng dụng đồ sộ hơn

#### Đặt máy chiếu dưới đất và áp dụng phản xạ gương

**Máy chiếu**

Hình 4‑5 Cách bố trí máy chiếu dưới đất kém theo gương

Với cách bố trí như hình Hình 4‑5, cơ bản cách bố trí này cho kết quả như là cách ở mục 4.4.1.2, nhưng có cải tiến về độ hiệu quả của của phương pháp chiếu. Việc áp dụng tính chất phản xạ của gương giúp ta có thể điều chỉnh độ rộng của vùng chiếu ra mà không cần phải tăng khoảng cách của mặt bàn và máy chiếu. Ngoài ra máy chiếu có thể tản nhiệt tốt hơn khi đặt nằm

Theo thực nghiệm, chúng em chọn góc đặt gương là góc 45o sẽ thi được hình ảnh to hơn trên màn hứng mà chất lượng ảnh vẫn không đổi

#### Kết luận

Theo qua trình phân tích và thực nghiệm ở cả 3 phương pháp trên, chúng em quyết định chọn phương pháp thứ 2: Đặt máy chiếu dưới đất theo phương vuông góc 90o để tiến hành xây dựng ứng dụng. Phương pháp 1 không mang lại trải nghiệm tốt do vấn đề đổ bóng. Phương pháp 2 và 3 gần tương đương nhau về thời gian lắp đặt và độ hiệu quả sử dụng. Tuy phương pháp 3 có thể tối ưu phần cứng của hệ thống hơn nhưng cần phải tính toán và thực nghiệm nhiều hơn. Nên chúng em sẽ chọn phương pháp 2.

Về bản chất, việc dùng máy chiếu và màn hứng để giả lập lại một màn chiếu tivi, hoặc màn hình cảm ứng kích thước lớn, nhưng do giá thành các thiết đó quá đắc đỏ, nên việc dùng phương pháp này sẽ giảm chi phí thực hiện đi rất nhiều lần.

### Kỹ thuật ánh xạ tọa độ giữa các không gian tọa độ

* Vấn đề

Với cấu hình hệ thống hiện tại như Hình 4‑1, camera sẽ ghi hình các thao tác của người dùng và chuyển ảnh đó vể module xử lý, máy chiếu sẽ chiếu ra hình ảnh sau khi được xử lý. Như vậy hiện đang tồn tại 2 hệ tọa độ:

Tọa độ trên ảnh ghi được bởi camera, chiều dài và chiều rộng bằng với kích thước mà camera trả về, đơn vị là pixel

Tọa độ trong môi trường Unity

Vấn đề đặt ra là làm sao ánh xạ tọa độ giữa 2 hệ trục tọa độ này

* Giải pháp

Qua quá trình nhận dạng vật thể, ta lấy được 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 4‑6 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 4‑6). 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 4‑7 Lấy mẫu Calibration 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 4‑7). 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:

### Kiến trúc hệ thống sự kiện liên hoàn

* Đặt vấn đề

Để có thể phát triển hệ thống theo hướng mở rộng, tích hợp, phát triển nhanh chóng các tính năng về sau, ta cần xây dựng một kiến trúc có tính linh hoạt cao, không bị phụ thuộc vào các thành phần con của hệ thống. Một ví dụ cụ thể như sau: giả sử sau này ta phát triển được một hệ thống nhận dạng chính xác, hiệu quả hơn hệ thống hiện tại, ta chỉ cần thay thế module nhận dạng mà không cần chỉnh sửa các phần còn lại của hệ thống mà vẫn đảm bảo được toàn hệ thống hoạt động bình thường.

* Giải pháp

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 4‑1



Sơ đồ 4‑1 Hệ thống sự kiện

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

Một hệ thống sự kiện liên hoàn có các thành phần sau :

* 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

#### Cấu trúc của một hệ thống sự kiện liên hoàn



Sơ đồ 4‑2 Hệ thống sự kiện liên hoàn

Sơ đồ 4‑2 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 4‑3). Nói cách khác đây là một Global Helper Class

****

Sơ đồ 4‑3 Sơ đồ lớp của Comparator

public class GreaterThanOrEqual : Comparator

{

public override bool compare(string FlagId, int value)

{

return Global.getFlag(FlagId) >= value;

}

}

Mã 1 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 4‑4)



Sơ đồ 4‑4 Sơ đồ lớp của Action

public class RotateObject : ActionType

{

public override void DoAction(GameObject gameObject, ParamsRaw paramsRaw)

{

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

}

}

Mã 2 Minh họa một Action

* **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.
* **Operator**: Mỗi hành động trong Post-actions là một Operator (cộng, trừ, nhân, chia, gán,…) tác động đến giá trị của Dictionary Flags như Sơ đồ 4‑5



Sơ đồ 4‑5 Sơ đồ lớp của Operator

public class Mod : Operator

{

public override int Operate(string FlagId, int value)

{

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

}

}

Mã 3 Minh họa một Operator

### Giao tiếp giữa các thiết bị tương tác

* Đặt vấn đề

Một hệ thống thường được chia làm nhiều module nhỏ khác nhau để dễ quản lý và thuận tiện cho việc bảo trì, nâng cấp về sau. Việc chia nhỏ các module làm phát sinh một vấn đề là vấn đề giao tiếp giữa chúng. Vậy các module đấy giao tiếp với nhau như thế nào, thông qua phương thức nào ?

* Giải pháp

Module A

Trung tâm xử lý và render

Module B

Nhận dạng vật thể

Module C

Tương tác nội dung với người dùng

Sơ đồ 4‑6 Cấu trúc module của hệ thống

Theo như kiến trúc ứng dụng trên được trình như ở Sơ đồ 4‑6, 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

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.

### Kiến trúc MVP trên các thiết bị di động

* Đặt vấn đề

Trước khi phát triển ứng dụng hỗ trợ trên thiết bị di động, điều chúng em quan tâm đến đầu tiên là việc chọn một giải pháp thiết kế kiến trúc hợp lí cho ứng dụng. Trong hệ thống, module (C) của chúng em được phát triển trên hệ điều hành Android. Do những tính chất đặc thù của hệ điều hành như vòng đời sống của các Activity, việc áp dụng một kiến trúc hợp lý là điều cần thiết

* Giải pháp

Các công cụ được cung cấp bởi android, với cách bài trí, và các cấu trúc dữ liệu, dường như hướng chúng ta theo mô hình Model View Controller (MVC) – đây cũng là mô hình chủ yếu chúng em được tiếp thu trong thời gian đại học. Thế nhưng MVC vẫn còn tồn tại những hạn điểm của nó vì chưa có sự tách bạch thật sự rõ ràng giữa các tầng. Chính vì thế trong hệ thống này chúng em quyết định dùng một cải tiến mới của mô hình MVC là mô hình MVP (Model View Presenter).



Hình 4‑8 Sơ đồ mô hình MVC

Nguồn <https://code.tutsplus.com/tutorials/an-introduction-to-model-view-presenter-on-android--cms-26162>



Hình 4‑9 Sơ đồ mô hình MVP

Nguồn : <https://code.tutsplus.com/tutorials/an-introduction-to-model-view-presenter-on-android--cms-26162>

MVP là một mô hình kiến trúc hướng giao diện người dùng được thiết kế để tạo thuận lợi cho việc kiểm thử unit test, và tăng tính tách biệt giữa tầng dữ liệu và tầng hiển thị dữ liệu (Hình 4‑8 và Hình 4‑9). Mô hình MVP cho phép tách tầng Presenter ra khỏi tầng Model, tương tác logic với giao diện được xử lí tách biệt với cách chúng ta biểu diễn trên tầng View. Hay nói cách khác, tầng View chỉ quan tâm xử lí việc nhận biết sự kiện tạo ra khi có sự thay đổi trên màn hình, còn mọi xử lí logic các sự kiện đó hay mọi xử lí ngầm cho việc giao tiếp với database hay API sẽ được gói trong tầng Presenter. Tầng View không cần quan tâm nhiều đến Model, mà chỉ nhận lệnh thay đổi thông qua Presenter hoặc gửi sự kiện tạo ra từ View đến Presenter để Presenter gửi xuống Model. Và với thiết kế này, lí tưởng nhất là cùng một logic ở tầng Presenter sẽ được áp dụng cho nhiều View khác nhau, và có thể hoán đổi được cho nhau. Các thành phần cơ bản có chức năng như sau:

* Model: Tầng xử lí dữ liệu, tầng này chịu trách nhiệm lấy giữ liệu hoặc update dữ liệu ở database hoặc network một cách bất đồng bộ. Sau đó sẽ trả về kết quả cho tầng Presenter thông qua các hàm callback.
* View: Tầng xử lí giao diện người dùng. Tầng này chịu trách nhiệm quản lí danh sách các view (bind view), đưa dữ liệu vào view, kiểm soát các animation, input event của user và gửi cho Presenter xử lí các event đó.
* Presenter: Tầng xử lí các business logic, đây là tầng trung gian duy nhất có thể giao tiếp với 2 tầng còn lại (2 tầng View và Model không giao tiếp được với nhau trực tiếp). Hoạt động thông thường của tầng này là nhận một input event từ tầng View, tầng Presenter sẽ lấy dữ liệu từ tầng Model gửi lại cho tầng View và hướng dẫn tầng View cách hiển thị.
* Như vậy ta có thể thấy được lợi điểm của MVP so với MVC truyền thống. Khi ở MVC, logic xử lí dữ liệu tương tác và xử lí dữ liệu hiển thị được gộp chung vào controller. Nên cả 2 tầng View và Controller đều có thể tương tác với người dùng, dẫn đến thiếu sự tách bạch trong chức năng các tầng. Chính nhờ sự phân biệt rõ ràng công việc của các tầng, MVP có thể chia một công việc lớn thành nhiều công việc nhỏ hơn, phân chia rành mạch các loại xử lí, dễ dàng bảo trì và mở rộng.

## Các chức năng của hệ thống

### Các cách thức tương tác

### Màn hình giao diện chính

## 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] J. Haas and B. Moriarty, “A History of the Unity Game Engine An Interactive Qualifying Project.”