ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM

BÁO CÁO

CHUYÊN ĐỀ MOBILE AND PERVASIVE COMPUTING

ĐỀ TÀI: LẬP TRÌNH GAME MOBILE TRÊN UNITY

GVHD: THẦY NGUYỄN TRÁC THỨC
LỚP: SE405.H11
SVTH: 12520347 – PHẠM MINH QUY
12520356 – HOÀNG HUY SƠN
12520395 – BÙI VĂN THÀNH
12520412 – ĐỖ ĐỨC THIỆN

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM

BÁO CÁO

CHUYÊN ĐỀ MOBILE AND PERVASIVE COMPUTING

ĐỀ TÀI: LẬP TRÌNH GAME MOBILE TRÊN UNITY

GVHD: THẦY NGUYỄN TRÁC THỨC
LỚP: SE405.H11
SVTH: 12520347 – PHẠM MINH QUY
12520356 – HOÀNG HUY SƠN
12520395 – BÙI VĂN THÀNH
12520412 – ĐỖ ĐỨC THIỆN

TP. Hồ Chí Minh, tháng 12 năm 2016

LỜI CẢM ƠN

Trên thực tế, không có sự thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ dù ít hay nhiều, dù trực tiếp hay gián tiếp của người khác. Trong suốt thời gian từ khi bắt đầu học tập tại trường đại học đến nay, chúng em đã nhận được rất nhiều sự quan tâm, giúp đỡ của quý Thầy Cô, gia đình và bạn bè.

Với lòng biết ơn sâu sắc nhất, chúng em xin gửi đến quý Thầy Cô ở Khoa Công nghệ Phần mềm – Trường Đại Học Công Nghệ Thông Tin đã cùng với tri thức và tâm huyết của mình để truyền đạt vốn kiến thức quý báu cho chúng em trong suốt thời gian học tập tại trường. Và đặc biệt, trong học kỳ này, Khoa đã tổ chức cho chúng em được tiếp cận với môn học mà theo chúng em là rất hữu ích đối với sinh viên ngành Công nghệ Phần mềm. Đó là môn học "Chuyên đề Mobile and Pervasive Computing".

Chúng em xin chân thành cảm ơn Thầy Nguyễn Trác Thức đã tận tâm hướng dẫn chúng em thông qua những buổi giảng dạy, nói chuyện và thảo luận. Nếu không có những lời hướng dẫn, dạy bảo của thầy thì chúng em nghĩ bài thu hoạch này sẽ rất khó có thể hoàn thiện được. Một lần nữa, chúng em xin chân thành cảm ơn thầy.

Sau cùng, chúng em xin kính chúc quý Thầy Cô trong Khoa Công nghệ Phần mềm và Thầy Nguyễn Trác Thức thật dồi dào sức khỏe, niềm tin để tiếp tục thực hiện sứ mệnh cao đẹp của mình là truyền đạt kiến thức cho thế hệ mai sau.

Trân trọng.

TP.HCM, ngày 20 tháng 12 năm 2016

Sinh viên thực hiện

Hoàng Huy Sơn

Phạm Minh Quy

Bùi Văn Thành

Đỗ Đức Thiện

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

MỤC LỤC

I. GIỚI THIỆU

1.1. Tổng quan về Mobile & Pervasive Computing

❖ Ubiquitous, Mobile, và Nomadic Computing

- Mobile Computing: "on-the-go", ví dụ, khi bạn đang ngồi trên một chuyến tàu, bạn vẫn có thể kết nối mạng như bình thường.
- Máy tính có ở hầu hết mọi nơi.

❖ Mobile Computing

- Mobile Computing đang ngày càng phát triển để đạt tới truyền dẫn không dây.
- Nhiều loại Mobile Computing đã được giới thiệu từ những năm 1990, bao gồm:
  - Personal Digital Assistant.
  - Enterprise Digital Assistant.
  - Smart phones.
  - UMPC.

❖ Mobile Computing Vision

- Những kết nối phổ biến – bất cứ nơi đâu, bất cứ lúc nào.
- Điều chỉnh sự đồng nhất của liên kết mạng lưới và truyền thông.
- Môi trường thông minh Ubiquitous – máy tính chạy hệ thống của chúng ở khắp mọi nơi.
- Tương tác người dùng dễ dàng.
- Truy cập độc lập với các dịch vụ và các thông tin phụ thuộc.

❖ Ubiquitous Computing (ubicomp)

- Ubicomp là một mẫu máy tính để bàn của con người nhằm tương tác với máy tính, trong đó quá trình xử lý thông tin đã được tích hợp một cách cẩn thận vào đối tượng và các hoạt động.

- Tích hợp máy tính để tương tác với toàn thế giới.
  - Invisible, máy tính ở khắp mọi nơi.
  - Thường được gọi là pervasive/invisible computing.
- Máy tính chủ yếu không phải là invisible, chúng chiếm ưu thế trong việc tương tác với con người.
- Ubicomp sẽ làm cho các máy tính trở nên invisible.
- Ubiquitous computing = mobile computing + môi trường thông minh.

Technology View

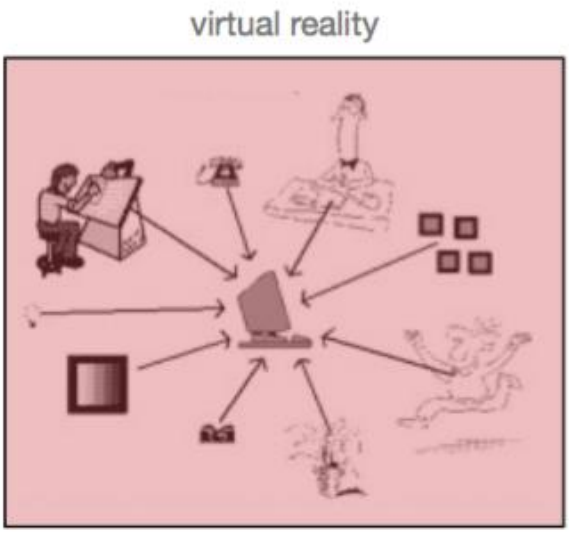
- Các hệ thống nhúng có ở khắp mọi nơi – tủ lạnh, máy giặt, ổ khóa, xe hơi, đồ nội thất.
- Môi trường thông minh.
- Các thiết bị di động và máy tính xách tay.
- Thông tin liên lạc không dây – điện thoại di động/cố định.

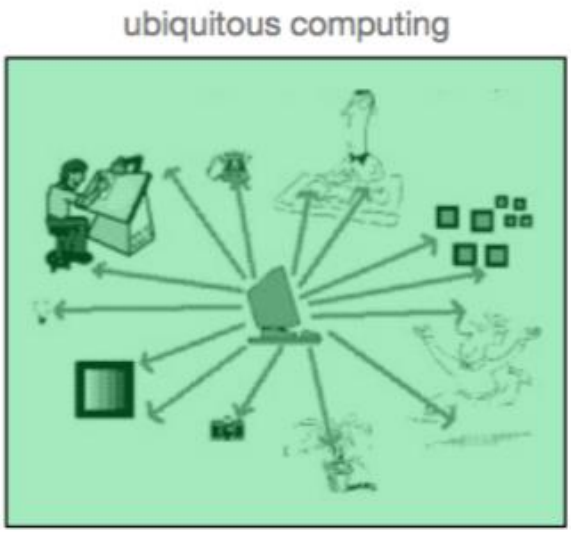
User View

- Invisible – tương tác ngầm với môi trường thiết bị của bạn.
- Tăng cường khả năng của con người trong tình huống xử lý các task.

❖ **Ubicomp vs Virtual Reality**

- Chúng ta có nên sống trong thế giới ảo hay không? Hay là nên loại bỏ máy tính và sống trong thế giới thực của chúng ta?
- VR là mô phỏng thế giới vật chất và đặt con người vào trong thế giới máy tính ảo (bị giới hạn các ứng dụng và hoạt động).
- Ubicomp là việc đưa máy tính đến với thế giới thực của con người, kết hợp các đối tượng và các hoạt động hằng ngày lại với nhau.
- Ubiquitous computing là sự kết hợp của yếu tố con người, khoa học máy tính, kỹ thuật và khoa học xã hội.

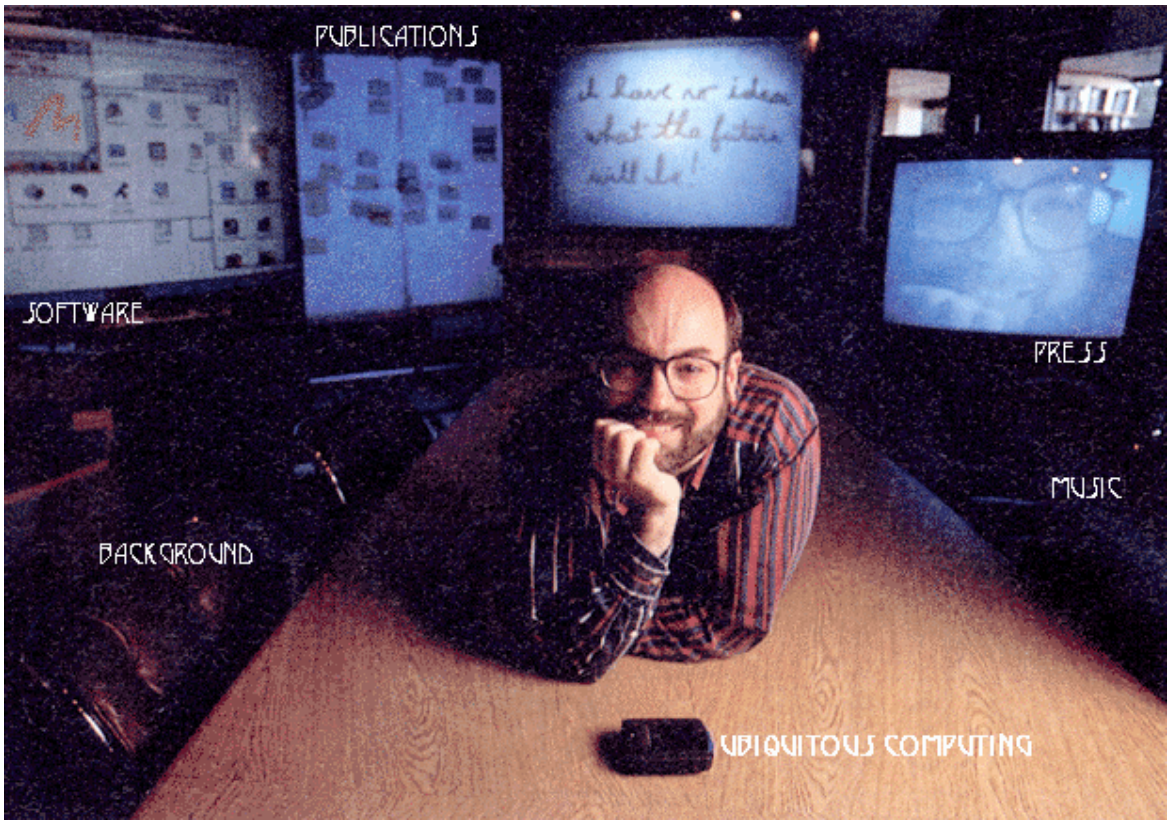




❖ **History of Ubicomp**

- Mark Weiser đặt ra cụm từ "Ubicomp computing" vào năm 1988, trong nhiệm kì nắm giữ chức vụ trưởng khoa công nghệ của trung tâm nghiên cứu Xerox Palo Alto.
- Cùng với giám đốc PARC và giám đốc khoa học John Seely Brown, Weiser đã viết một số bài báo đầu tiên về đề tài này, chủ yếu là xác định và phác thảo những mối quan tâm lớn nhất về nó.
- Andy Hopper từ Đại học Cambridge vương quốc Anh đã đề xuất và chứng minh khái niệm "Teleport" - ứng dụng theo dõi người dùng bất cứ khi nào họ di chuyển.
- Bill Schilit (nay làm việc tại Google) cũng đã thực hiện một số công việc trước đó trong vấn đề này, và tham gia vào các hội thảo về Mobile Computing được tổ chức tại Santa Cruz năm 1996.
- Tiến sĩ Ken Sakamura của Đại học Tokyo, Nhật Bản dẫn đầu phòng thí nghiệm Ubiquitous Networking (UNL), Tokyo cũng như diễn đàn T-Engine.
- Mục tiêu chung của Sakamura's Ubiquitous Networking và diễn đàn T-Engine là cho phép bất kì thiết bị nào cũng có thể phát và nhận thông tin mỗi ngày.

- Roy Want, trong khi đang là một nhà nghiên cứu và sinh viên đang làm việc dưới quyền Andy Hopper tại Đại học Cambridge, đã làm việc trên "Active Badge System", một hệ thống xác định vị trí tiên tiến, nơi mà người dùng di động cá nhân đã được sát nhập với máy tính.
- MIT cũng đã đóng góp những nghiên cứu quan trọng về lĩnh vực này, đáng chú ý là tập đoàn Things That Think tại Media Lab và những nỗ lực của CSAIL vào Project Oxygen.
- Những đóng góp quan trọng khác bao gồm University of Washington's Ubicomp Lab, Georgia Tech's College of Computing, Cornell University's People Aware Computing Lab, NYU's Interactive Telecommunications Program, UC Irvine's Department of Informatics, Microsoft Research, Intel Research.



Mark Weiser, cha đẻ của Ubicomp

❖ **Ubicomp core concepts**

- Ubiquitous computing (ubicomp) là một khái niệm điện toán tiên tiến, nơi mà máy tính có thể xuất hiện ở bất cứ khi nào và bất cứ nơi đâu.
- Ngược lại với máy tính để bàn, ubiquitous computing có thể xuất hiện bằng việc sử dụng bất kì thiết bị, bất kì vị trí và bất kì định dạng nào.
- Một người dùng tương tác với máy tính, có thể tồn tại ở nhiều dạng khác nhau, bao gồm máy tính xách tay, máy tính bảng, thiết bị đầu cuối và điện thoại.
- Công nghệ cơ bản hỗ trợ ubiquitous computing bao gồm Internet, các phần mềm trung gian tiên tiến, hệ điều hành, mã điện thoại, cảm biến, bộ vi xử lý, I/O mới, giao diện người dùng, mạng, các giao thức điện thoại di động, vị trí, định vị, và ngay cả những vật liệu mới.
- Ví dụ, một môi trường ubiquitous computing trong nước có thể kết nối ánh sáng và các môi trường điều khiển với màn hình sinh trắc học cá nhân để có thể tạo nên những bộ quần áo, chiếu sáng và sưởi ấm trong điều kiện một căn phòng có thể được điều chế, liên tục và không đáng kể.
- Một viễn cảnh khác là tủ lạnh có thể "nhận thức" các nội dung đã được lập trình trên chúng, chúng ta có thể cầm một loạt các loại thực phẩm trên tay và tủ lạnh sẽ cảnh báo người dùng những thực phẩm đã cũ hoặc bị hư hỏng.
- Ubiquitous computing là những thách thức trên khoa học máy tính: trong thiết kế hệ thống và kỹ thuật, xây dựng mô hình hệ thống, và trong thiết kế giao diện người dùng.
- Mô hình hiện đại của tương tác giữa con người và máy tính, cho dù dòng lệnh, menu điều khiển, hoặc dựa trên GUI, không phù hợp và không đủ để giải quyết các trường hợp phổ biến.

- Ubiquitous computing có thể được nhìn thấy bao gồm nhiều lớp, mỗi lớp với vai trò của mình, kết hợp lại với nhau tạo thành một hệ thống duy nhất.
- Lớp 1: Lớp quản lý công việc:
  - Nhiệm vụ của người sử dụng cần cho các dịch vụ trong môi trường.
  - Quản lý những thuộc tính phức tạp.
- Lớp 2: Lớp quản lý môi trường:
  - Để theo dõi một nguồn lực và khả năng.
- Lớp 3: Lớp môi trường:
  - Để theo dõi một nguồn tài nguyên có liên quan.
  - Để quản lý tin cậy của các nguồn tài nguyên.

❖ **Intelligence**

- Máy tính nhúng để tăng cường đối tượng vật lý.
- Đạt được trí thông minh thông qua kết nối của các đối tượng vật lý.
- Đạt được trí thông minh thông qua nhận thức vị trí (không có AI). Ví dụ: Chuyển tiếp cuộc gọi tự động (context awareness) điều khiển ánh sáng → tường cảm biến thông minh sưởi ấm → kiểm soát ánh sáng.

❖ **Early work**

Tabs:

- Rất nhỏ - huy hiệu thông minh với thông tin người dùng, lịch, nhật ký,…
- Cho phép thiết lập cá nhân để theo dõi người dùng.
- Tiến hành xung quanh bởi một người.

❖ **Current Trends**

Touch Pads:

- Foot-scale Ubicomp devices:
  - Tờ giấy/ máy tính bảng.
  - Máy tính xách tay nhưng không phải laptop.
- Tens in a room:
  - Giống như giấy phế liệu có thể được nắm lấy và sử dụng bất cứ nơi nào, không có ID đặc trưng.

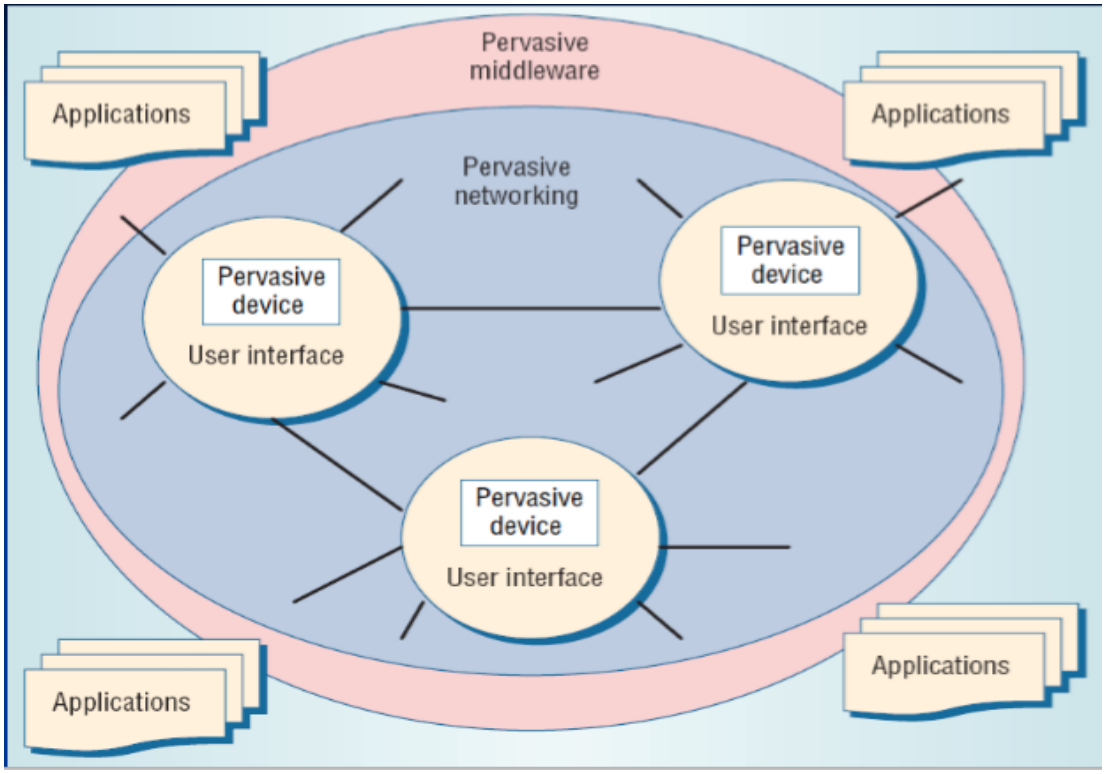
❖ **Current Technology**

- Thiết bị thông tin di động:
  - Laptops, notebooks, and sub-notebooks.
  - Máy tính xách tay.
  - PDAs và điện thoại thông minh.
- Mạng lưới thông tin liên lạc không dây:
  - Nhiều mạng bao phủ toàn cầu.
- Internet:
  - TCP / IP và các giao thức ứng dụng de-facto.

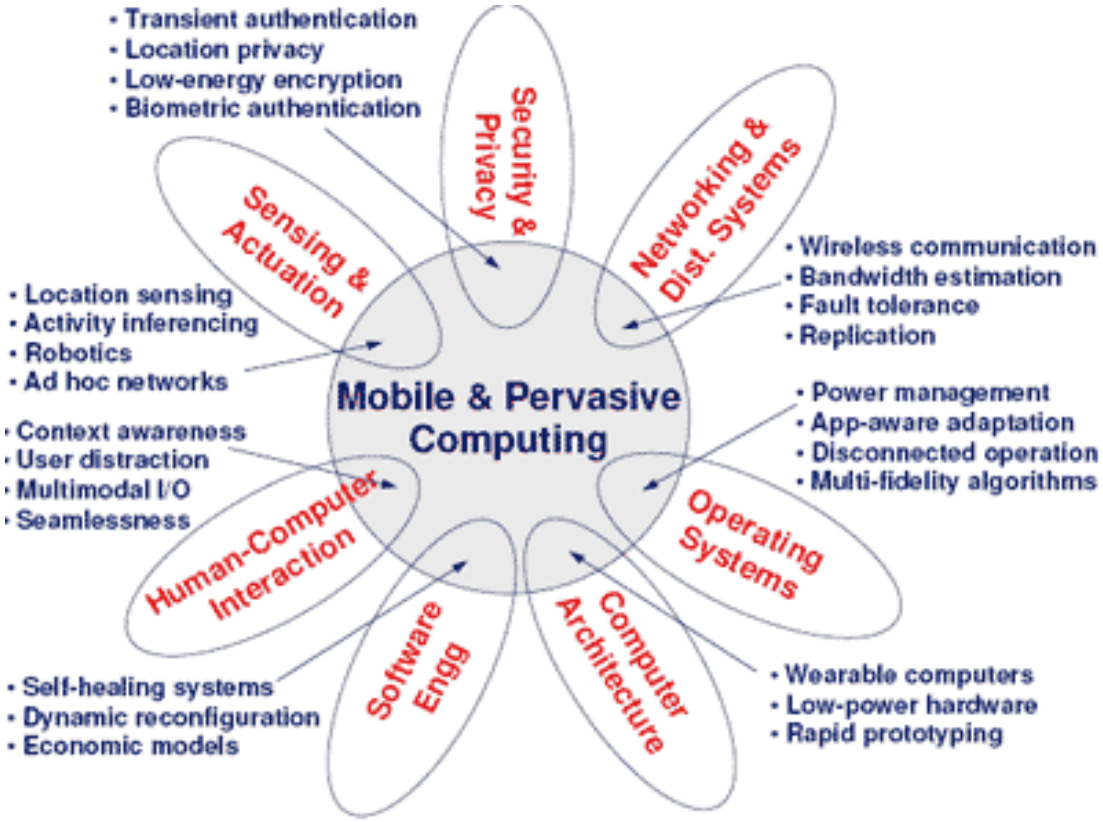
❖ **Usability**

- Giao diện người dùng chung cho các máy trạm và các ứng dụng thiết bị di động.
- Hiển thị thông tin thay đổi.
- Nhận dạng giọng nói + văn bản để chuyển thành hội thoại.
- Nhận diện cử chỉ.
- Intelligent agents.

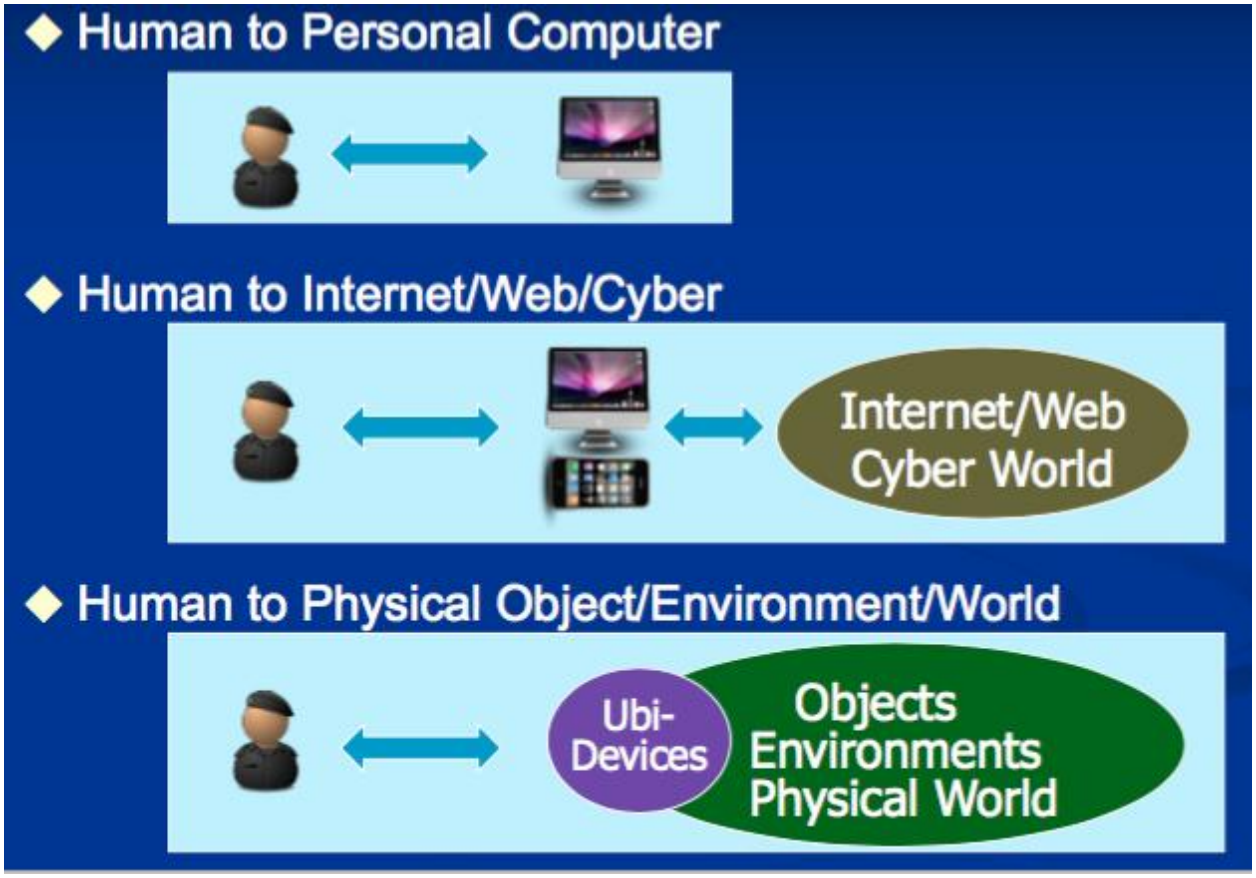
❖ **Ubiquitous/Pervasive Computing Framework**



❖ **Related Fields**



❖ **Human-Computer Interaction**



❖ **Integration of Mobile Systems**

- Không để thiết bị mất kết nối: Cần tương tác với các hệ thống thông tin phức tạp. Ví dụ: Cơ sở dữ liệu lớn – sáp nhập thông tin cập nhật, bảng hiển thị,…
- Phát triển hệ thống:
  - Yêu cầu đặc điểm kỹ thuật cho các hệ thống thích nghi.
  - Thành phần để đáp ứng QoS toàn cầu, an ninh, yêu cầu độ tin cậy và hiệu suất.
- Mô hình di động:
  - Đặc điểm kỹ thuật và phân tích hành vi.
  - Mô hình bối cảnh hệ thống nhận biết.

❖ **Context Aware Computing**

- Nó là một khái niệm phổ biến trong tương tác giữa người và máy tính.
- Mục tiêu của context-aware computing là tiếp thu và sử dụng thông tin về bối cảnh của một thiết bị để cung cấp các dịch vụ thích hợp cho riêng người, địa điểm, thời gian, sự kiện,...
- Ví dụ, một điện thoại di động sẽ luôn luôn rung và không bao giờ kêu bíp trong một buổi hòa nhạc, nếu hệ thống có thể biết được vị trí của điện thoại di động và lịch trình buổi hòa nhạc.

❖ **Context Adaptation**

- Một hệ thống context adaptive cho phép người sử dụng dùng để duy trì một số ứng dụng (trong các hình thức khác nhau) trong khi chuyển vùng giữa các công nghệ khác nhau như truy cập không dây, địa điểm, thiết bị và thậm chí đồng thời thực hiện các công việc hàng ngày như các cuộc họp, lái xe,...

❖ **Issues : Context Awareness**

- Vị trí hiện tại: Dùng để xác định vị trí, ví dụ: GPS – radio beacon, IR,…
- Hoạt động người dùng: Đi bộ, lái xe, đi xe buýt - làm thế nào để phát hiện điều này?
- Môi trường xung quanh: Trong rạp chiếu phim, nhà hát, ở một mình.
- Tài nguyên hoặc các dịch vụ có sẵn: Khả năng của thiết bị.
- Màn hình, thiết bị đầu vào, sức mạnh xử lý, tuổi thọ pin,…
- QoS hiện tại sẵn có - đặc biệt là cho các liên kết vô tuyến.

❖ **Intelligent Environment**

- Một môi trường thông minh là một vị trí (ví dụ: nhà, văn phòng, bệnh viện, ...) được trang bị cảm biến, thiết bị truyền động và các máy tính được nối mạng với nhau và với internet.
- Các thành phần được điều khiển "Cảnh báo thông minh" bởi phần mềm.
- Nó điều chỉnh môi trường cho phù hợp với con người.
- Con người có thể nói chuyện với các môi trường bằng việc sử dụng lời nói và ngôn ngữ tự nhiên và các cảm biến có thể giám sát môi trường.

❖ **Smart Dust**

- "Smart Dust" là thiết bị rất nhỏ không dây Micro Electro Cơ Sensors (MEMS), có thể phát hiện tất cả mọi thứ từ ánh sáng đến rung động.
- Cảm biến nhiệt độ, độ ẩm, ánh sáng, chuyển động .... với đài phát thanh hai chiều hoặc laser + pin.
- Các ứng dụng tiêu biểu:
  - Liên quan đến quốc phòng cảm biến ở chiến trường, phát hiện chuyển động,…
  - Giám sát chất lượng của sản phẩm - độ rung, độ ẩm, độ nóng.
  - Giám sát các thành phần của xe hơi,…

❖ **Issues**

- Pin sẽ là nguồn năng lượng không thực tế cho 100K vi xử lý cho mỗi người.
- Năng lượng mặt trời là không thích hợp cho tất cả các môi trường.
- Sức mạnh không phải là tốc độ là vấn đề quan trọng cho các thiết kế bộ vi xử lý trong tương lai.

❖ **Major Challenges**

Hardware Prototype Issue:

- Công suất tiêu thụ: Không thể thay pin cho nhiều thiết bị Ubicomp thường xuyên.
- Cân bằng tính năng HW/ SW: Màn hình, mạng, xử lý, bộ nhớ, khả năng lưu trữ, đa nhiệm, QoS,...
- Dễ mở rộng và sửa đổi (tích hợp so với mô-đun).

Network Issue:

- Wireless Media Access (802.11, Bluetooth, Cellular Networks).
- Quality of Services (RSVP, etc.).
- Ubicomp devices thay đổi điểm gắn mạng. (Mobile IP).

Application Issue:

- Locating people (active badges).
- Shared drawing in virtual meeting.

❖ **Security**

- Tương tác sẽ được qua nhiều ranh giới tổ chức đặc điểm kỹ thuật, phân tích và tích hợp cho hệ điều hành không đồng nhất, cơ sở dữ liệu, tường lửa, router.
- Mọi thứ có khả năng bị hack.
- Là máy truyền động nhỏ, với các dữ liệu bí mật, có thể dễ dàng bị mất hoặc bị đánh cắp - xác thực sinh trắc học.
- Tồn tại công nghệ bảo mật cần thiết.

❖ **Privacy**

- Dịch vụ định vị theo dõi chuyển động với đơn vị metres.

❖ **Management**

- Khổng lồ, hệ thống phức tạp:
  - Hàng tỉ vi xử lý.
  - Nhiều tổ chức.
  - Quản lý thế giới vật lý, kiểm soát các cảm biến, thiết bị truyền động.
- Thiên đường của hacker và virus.
- Hệ thống truyền thông tin sai lệch về các cá nhân hoặc tổ chức.
- Sự phức tạp của s / w cài đặt trên máy trạm hay máy chủ - làm thế nào để bạn đối phó với hàng tỷ vấn đề?

1.2. Giới thiệu về Project sẽ phát triển

Thiết bị di động đang trở thành một phần thiết yếu của cuộc sống. Smartphone, Tablet đang dần vượt mặt PC về độ thông dụng và số lượng người dùng. Chính bởi vậy, phát triển ứng dụng di động trở thành một nghề hấp dẫn.

Trước doanh thu hàng triệu USD/ngày của những tựa game như Clash of Clans, Puzzle & Dragons, I am MT… không ít studio đã chuyển hướng tập trung vào phát triển trò chơi cho smartphone. Cùng với đó là sự lên ngôi của những engine game như BlitzTech, GameBryo, Havok Visio… và đặc biệt phải kể tới Unity.

Unity là một engine game đa nền tảng được phát triển bởi Unity Technologies. Engine này được phát triển bằng C/C++ và có khả năng hỗ trợ mã viết bằng C#, JavaScript hoặc Boo. Hiện tại, Unity đã phát triển tới phiên bản thứ 4, đồng thời là sự lựa chọn số 1 cho các studio game. Những ưu thế lớn nhất của Unity có thể kể tới: hiệu năng cao, chi phí hợp lý (nếu không muốn nói là rẻ), hỗ trợ đa nền tảng (gần như mọi nền tảng hiện nay như PlayStation 3, Xbox 360, Wii U, iOS, Android, Windows…) và sự ổn định đáng kinh ngạc.

Nhằm góp phần làm phong phú kho tàng game của người dùng cũng như tăng sự lựa chọn giải trí cho mọi người và đặc biệt để thỏa mãn đam mê làm game của cá nhân, nhóm đã lên ý tưởng để làm ra một game casual đơn giản với tên gọi "JustJump". Game được xây dựng với lối chơi không có điểm kết thúc mà chỉ tăng dần độ khó. Với những hình ảnh và âm thanh sống động, game hứa hẹn sẽ mang đến cho người dùng những trải nghiệm hoàn toàn mới lạ và những giây phút thư giãn thoải mái.

II. CÔNG NGHỆ

2.1. Cảm biến trên Mobile

❖ Con quay hồi chuyển

- **Con quay hồi chuyển (Gyroscope)** là một lớp tính năng trong UnityEngine.
- Sử dụng lớp này để giải quyết các vấn đề liên quan tới cảm biến (bằng con quay hồi chuyển tích hợp trong smartphone) trong lập trình game bằng Unity.
- Bao gồm các thành phần thuộc tính:
  - **attitude:** Hướng hiện tại của thiết bị.
  - **enabled:** Trạng thái tắt mở của gyroscope.
  - **gravity:** Trả về một vector gia tốc của trọng lực thể hiện trong hệ quy chiếu của thiết bị.
  - **rotationRate:** Trả về tốc độ quay được đo bằng cảm biến của thiết bị. rotationRate thể hiện cho tốc độ quay xung quanh một trong ba trục tọa độ với đơn vị tính bằng rad/s. Đây là giá trị khi nó được thể hiện bởi phần cứng cảm biến.

❖ **Cảm biến gia tốc**

- Chúng ta sử dụng gia tốc thông qua struct **AccelerationEvent** trong UnityEngine.
- Khi thiết bị di chuyển thì giá trị gia tốc sẽ được trả về theo 3 hệ trục trong không gian theo trong lực thức tế. Giá trị sẽ nhận từ -1 đến 1. Giá trị càng gần bằng 1 thì có nghĩa thiết bị đang hướng về phía đó càng cao.
- Nếu bạn để điện thoại đứng (nút home ở phía dưới) đối diện với mình thì trục X sẽ chiều từ trái qua phải, trục Y từ dưới đất lên và trục Z sẽ hướng về bạn.
- Struct AccelerationEvent có hai thành phần thuộc tính:
  - **acceleration:** Giá trị của gia tốc.
  - **deltaTime:** Khoảng thời gian giữa hai lần cập nhật gia tốc.

2.2. Touch

❖ **Tổng quan về Touch trong Unity**

- **MultiTouch** trong Unity được xác định là một mảng các Touch.
- **Touch**:
  - Cấu trúc mô tả các trạng thái của ngón tay khi chạm vào màn hình.
  - Thiết bị có thể theo dõi một số phần khác nhau của dữ liệu về một sự va chạm trên màn hình cảm ứng, bao gồm các trạng thái của nó (bắt đầu va chạm, kết thúc hoặc di chuyển), vị trí và liệu sự va chạm này là va chạm đơn hay còn một vài thao tác khác nữa. Hơn nữa, sự va chạm liên tiếp giữa các bản cập nhật frame có thể được phát hiện bởi thiết bị, do đó, một số ID phù hợp có thể được thể hiện thông qua các frame và được sử dụng để xác định làm thế nào để biết một ngón tay đang chuyển động.

- Cấu trúc Touch được sử dụng bởi Unity dùng để lưu trữ các dữ liệu liên quan đến trường hợp va chạm duy nhất và được trả về bởi hàm Input.GetTouch. GetTouch sẽ yêu cầu phải cập nhật trên mỗi bản frame để có được những thông tin cảm ứng mới nhất từ thiết bị và thuộc tính fingerId có thể được sử dụng để xác định cách thức va chạm giữa các frame.

❖ **Các thành phần thuộc tính của Touch**

- **altitudeAngle**: Giá trị 0 radian chỉ ra rằng vật chạm song song với bề mặt, pi / 2 chỉ ra đó là góc vuông.
- **azimuthAngle**: Giá trị 0 radian chỉ ra rằng vật chạm nằm dọc theo trục x của thiết bị.
- **deltaPosition**: Vị trí delta kể từ lần thay đổi cuối cùng. Vị trí tuyệt đối của các va chạm được ghi lại một cách định kỳ và có sẵn trong thuộc tính position. Giá trị deltaPosition là một Vector 2 chiều thể hiện cho sự khác biệt giữa các vị trí va chạm được ghi lại trên những bản cập nhật gần đây nhất và các bản cập nhật trước đó. Chúng ta cũng có thể tính toán tốc độ va chạm của chuyển động bằng cách chia deltaPosition.magnitude cho deltaTime.
- **deltaTime**: Khoảng thời gian đã trôi qua kể từ lần thay đổi giá trị cuối cùng được ghi lại trong Touch.
- **fingerId**: Chỉ số duy nhất cho các va chạm. Tất cả các va chạm đều được thể hiện trong mảng Input.touches. Giá trị ID này rất hữu ích khi phân tích các cử chỉ và trở nên tin cậy hơn trong việc xác định các ngón tay bởi các vị trí trước đó của chúng khá gần nhau.
- **position**: Vị trí của các va chạm tính theo tọa độ pixel.
- **pressure**: Áp lực đè cho một va chạm. 1.0f được xem là áp lực trung bình của một va chạm.
- **radius**: Giá trị ước lượng bán kính của một va chạm.

- **tapCount**: Số va chạm. Khi tay chạm vào màn hình, thì sẽ xác định là 1 touch, khi có thêm 1 touch nữa lên màn hình, thì số touch sẽ là 2 v.v .... Tuy nhiên, khi một trong 2 touch được nhả ra, thì touch còn lại được xem là touch thứ nhất.
- **type**: Giá trị cho biết va chạm thuộc loại Direct (ngón tay), Indirect (qua remote) hay Stylus (bút chạm).
- **phase**: Giai đoạn của một cái chạm. Các phase này thuộc kiểu **TouchPhase.**
- Các loại **TouchPhase**:
  - **Began**: Lúc mới touch, chạm vào màn hình.
  - **Moved**: Di chuyển ngón tay trên màn hình.
  - **Stationary**: Ngón tay chạm vào màn hình nhưng không di chuyển.
  - **Ended**: Ngón tay thoát khỏi màn hình. Đây là giai đoạn cuối cùng của sự va chạm.
  - **Canceled**: Hệ thống sẽ hủy cho việc theo dõi sự va chạm. Điều này có thể xảy ra nếu người dùng đặt thiết bị lên mặt của họ hoặc đồng thời thực hiện nhiều chạm hơn so với hệ thống có thể theo dõi (con số chính xác khác nhau với các nền tảng khác nhau).

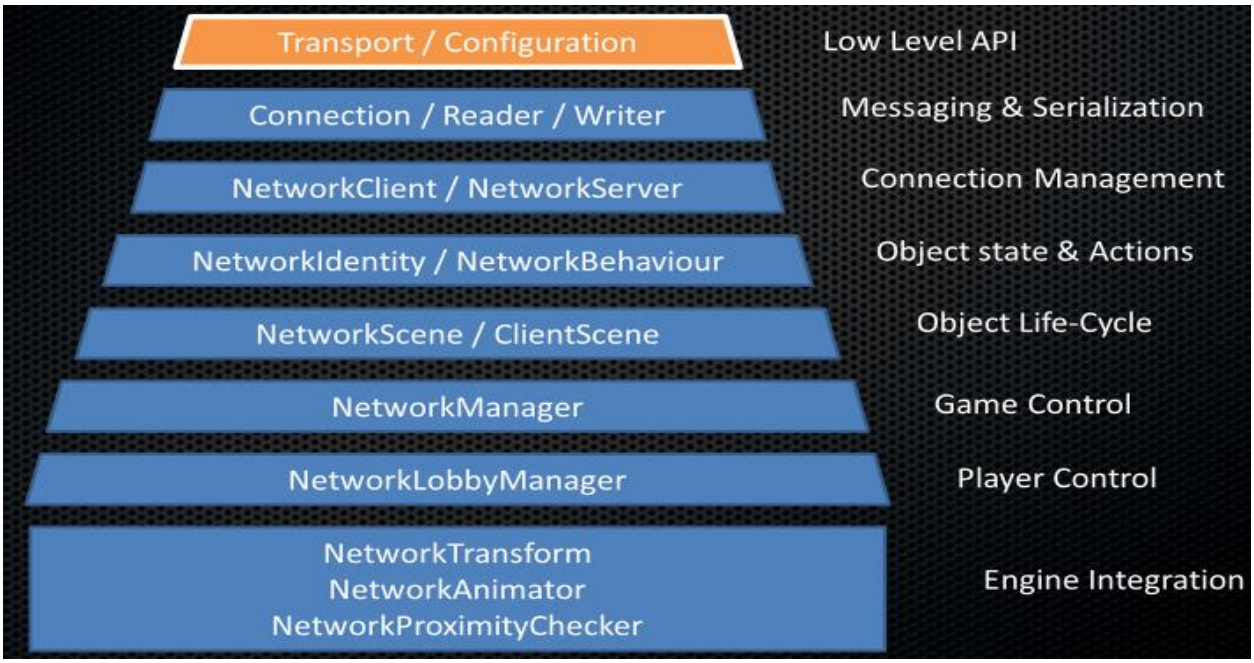
2.3. Networking

❖ Tổng quan về Networking trong Unity

- The High Level API (HLAPI) là một hệ thống để xây dựng các trò chơi multiplayer Unity.
- HLAPI chứa một tập hợp các lệnh được xây dựng để tạo Unity Networking, và chúng được chứa bên trong UnityEngine.Networking. Tập trung vào tính dễ sử dụng, phát triển lặp và cung cấp các dịch vụ hữu ích cho nhiều trò chơi. Chẳng hạn như:

- Xử lý tin nhắn.
- Đồng bộ trạng thái.
- Quản lý phân phối đối tượng.
- Lớp mạng: Server, Client, Connection,…

❖ **Các thành phần của Unity Networking**



Mô hình Unity Networking

- **NetworkReader:** Là một lớp API cấp cao cho việc đọc các đối tượng từ các luồng byte. Lớp này làm việc cùng với NetworkWriter.NetworkReader có chức năng tuần tự cụ thể đối với nhiều loại Unity.
- **NetworkWriter:** Là một lớp API cấp cao để viết các đối tượng để các luồng byte. Lớp này làm việc cùng với NetworkReader. NetworkWriter có chức năng tuần tự cụ thể đối với nhiều loại Unity. Các NetworkWriter có thể được sử dụng với các lớp MessageBase để làm cho các mảng byte chứa đăng tin nhắn mạng.

- **NetworkBehaviours:** Là những Script đặc biệt làm việc với các đối tượng với các thành phần NetworkIdentity. Những Script này có thể thực hiện chức năng HLAPI như lệnh: ClientRPCs, SyncEvents và SyncVars.
- **NetworkClient:** Là một lớp HLAPI quản lý kết nối mạng đến một máy chủ - trong trường hợp này là một NetworkServer. Nó có thể được sử dụng để gửi tin nhắn đến máy chủ và nhận tin nhắn từ máy chủ. Các NetworkClient cũng giúp quản lý các đối tượng mạng sinh ra, và định tuyến thông điệp RPC và các sự kiện mạng.
- **NetworkIdentity:** Là một thành phần Unity và là trung tâm của hệ thống mạng mới. Nó có thể được thêm vào các đối tượng từ AddComponents → Network → NetworkIdentity ở thanh menu. thành phần này điều khiển trạng thái trên mạng của một đối tượng, và nó làm cho các hệ thống mạng nhận biết được được nó.
- **NetworkManager:** Là một lớp cao hơn, cho phép bạn kiểm soát trạng thái của một trò chơi mạng. Nó cung cấp một giao diện trong trình soạn thảo để kiểm soát cấu hình của mạng, các prefabs dùng để sinh sản, và những cảnh để sử dụng cho các trạng thái mạng khác nhau của trò chơi.
- **Network Manager HUD:** Cung cấp một giao diện người dùng mặc định cho việc kiểm soát tình trạng mạng của trò chơi. Nó cũng hiển thị thông tin về trạng thái hiện tại của NetworkManager trong trình soạn thảo.

- **Network Lobby Manager**: Là một loại của NetworkManager cung cấp một lobby cho nhiều người trước khi vào cảnh chơi chính của trò chơi. Nó cho phép bạn thiết lập một mạng lưới với:
  - Một giới hạn người chơi tối đa.
  - Tự động bắt đầu khi tất cả người chơi sẵn sàng.
  - Cho phép chặn người chơi.
  - Nhiều tùy biến cho người chơi chọn khi ở trong Lobby.
- Có 2 loại đối tượng người chơi trong NetworkLobbyManager:
  - Lobby Player Object:
    - Một cho mỗi người chơi.
    - Được tạo ra khi client kết nối.
    - Tồn tại cho đến khi client ngắt kết nối.
    - Lưu cấu hình dữ liệu.
  - Game Player Object:
    - Một cho mỗi người chơi.
    - Tạo ra khi cảnh trò chơi được bắt đầu.
    - Phá hủy lobby khi nhập lại.
    - Xử lý các lệnh trong game.
- **NetworkTransform** đồng bộ hóa chuyển động của đối tượng trò chơi trên mạng. thành phần này có thể truy cập vào tài khoản, do đó đối tượng LocalPlayer đồng bộ hóa vị trí của họ từ máy khách đến máy chủ, sau đó ra cho các khách hàng khác. Các đối tượng khác (với quyền máy chủ) đồng bộ hóa vị trí của họ từ máy chủ cho khách hàng. Để sử dụng thành phần này, thêm nó vào đối tượng prefab hay trò chơi mà bạn muốn đồng bộ phong trào. Các thành phần yêu cầu các đối tượng trò chơi có một NetworkIdentity. Lưu ý rằng các đối tượng mạng phải được sinh ra để đồng bộ hóa.
- **NetworkAnimator**: dùng để đồng bộ hóa hình ảnh động trên Network.

- **NetworkProximityChecker** là một thành phần có thể được sử dụng để kiểm soát khả năng hiển thị của các đối tượng client cho mạng, dựa trên sự gần gũi với người chơi mạng. Để sử dụng nó, đặt các thành phần trên các đối tượng đó phải được nhìn thấy trong một phạm vi nhất định. Lớp này dựa trên vật lý để tính toán tầm nhìn.

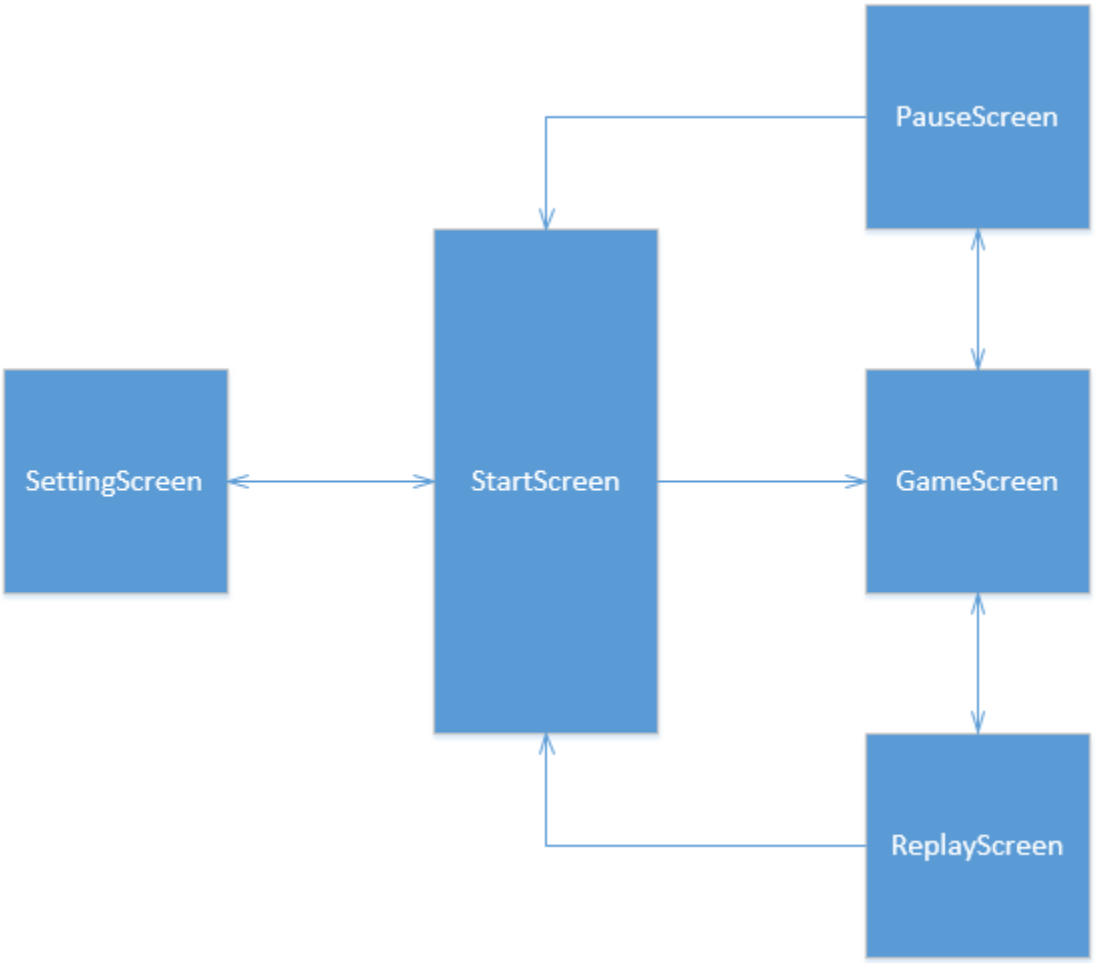
2.4. Tối ưu Game Mobile làm bằng Unity

Theo bài viết **Performance tips for Unity 2d mobile (**divillysausages.com – Một trang web chuyên về thủ thuật lập trình) chúng ta có một số cách đơn giản để tối ưu Game Mobile làm bằng Unity như sau:

- Thiết lập Application.targetFrameRate lên 60 thay vì mặc định cho điện thoại là 30. Điều này sẽ làm game mượt hơn rất nhiều lần.
- Khi kiểm thử game thì nên thử trên thiết bị di động. Bởi khi chạy trên desktop thì có thể bạn sẽ thấy nó không có lỗi nhưng bạn không thể biết điều gì sẽ xảy ra khi có một tap chạm vào game.
- Sử dụng TexturePacker để gom resource thành một file duy nhất. Điều này sẽ làm giảm sự rời rạc của tài nguyên game.
- Bỏ những hàm rỗng trong game. Mặc dù chúng rỗng nhưng nếu chúng ta để thì khi có nhiều hàm như vậy trong một game lớn thì chắc chắn chúng ta sẽ làm chậm game ở một khoảng thời gian nhất định nào đó.
- Lưu giữ những biến cần thiết cho các thành phần thường sử dụng. Tránh khởi tạo chúng thường xuyên, điều này sẽ mất thời gian rất nhiều. Đặc biệt là trong các hàm Update / FixedUpdate.

III. THIẾT KẾ VÀ CÀI ĐẶT

3.1. Kiến trúc hệ thống



Mô hình chuyển đổi giữa các Screen

a. SettingScreen

- SettingScreen là màn hình cài đặt của game.
- Một số đối tượng chính của màn hình này bao gồm:

Tên	Kiểu	Mô tả
ok setting button	Button	Dùng để trở về màn hình StartScreen
music volume	Slider	Dùng để tăng/giảm nhạc nền của game
sound volume	Slider	Dùng để tăng/giảm âm thanh của toàn bộ game
music text	Text	Dùng để hiển thị "Music Volume"
sound text	Text	Dùng để hiển thị "Sound Volume"

b. StartScreen

- StartScreen là màn hình bắt đầu trước khi vào chơi game.
- Một số đối tượng chính của màn hình này bao gồm:

Tên	Kiểu	Mô tả
start button	Button	Dùng để bắt đầu vào chơi game
setting button	Button	Dùng để đi đến SettingScreen
credit button	Button	Dùng để đi đến CreditScreen

c. PauseScreen

- PauseScreen là màn hình để tạm dừng chơi game.
- Một số đối tượng chính của màn hình này bao gồm:

Tên	Kiểu	Mô tả
resume icon	Icon	Icon thể hiện đang ở chế độ Pause
resume button	Button	Dùng để quay lại GameScreen
back menu	Button	Dùng để đi đến StartScreen

d. GameScreen

- GameScreen là màn hình chính để chơi game.
- Một số đối tượng chính của màn hình này bao gồm:

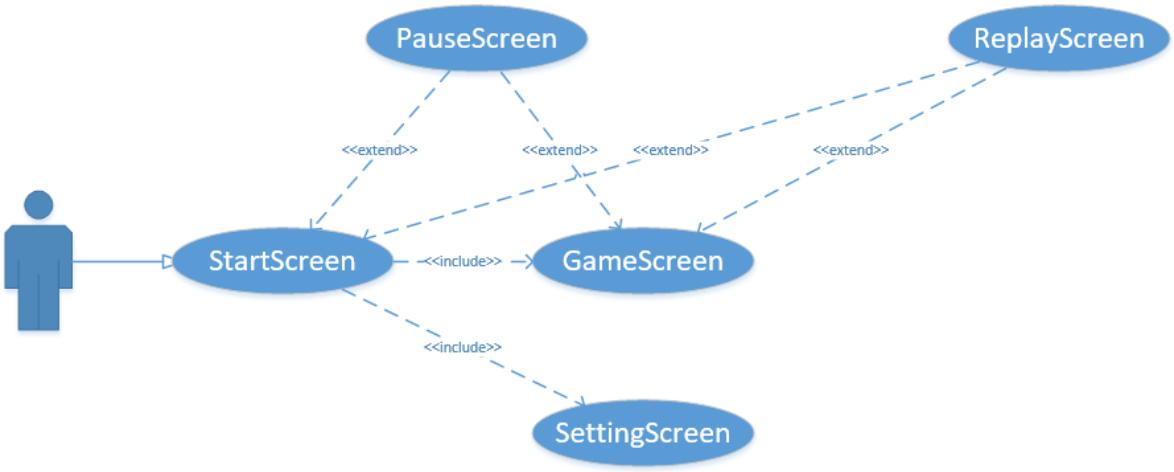
Tên	Kiểu	Mô tả
land	GameObject	Đối tượng land
jumper	GameObject	Đối tượng nhân vật
pause button	Button	Dùng để đi đến StartScreen
land image	Image	Hình ảnh của land
landcount	Text	Hiển thị số land đã nhảy qua
scoreimage	Image	Hình ảnh của score
score	Text	Hiển thị số điểm đạt được

e. ReplayScreen

- ReplayScreen là màn hình để chơi lại game sau khi đã bị chết.
- Một số đối tượng chính của màn hình này bao gồm:

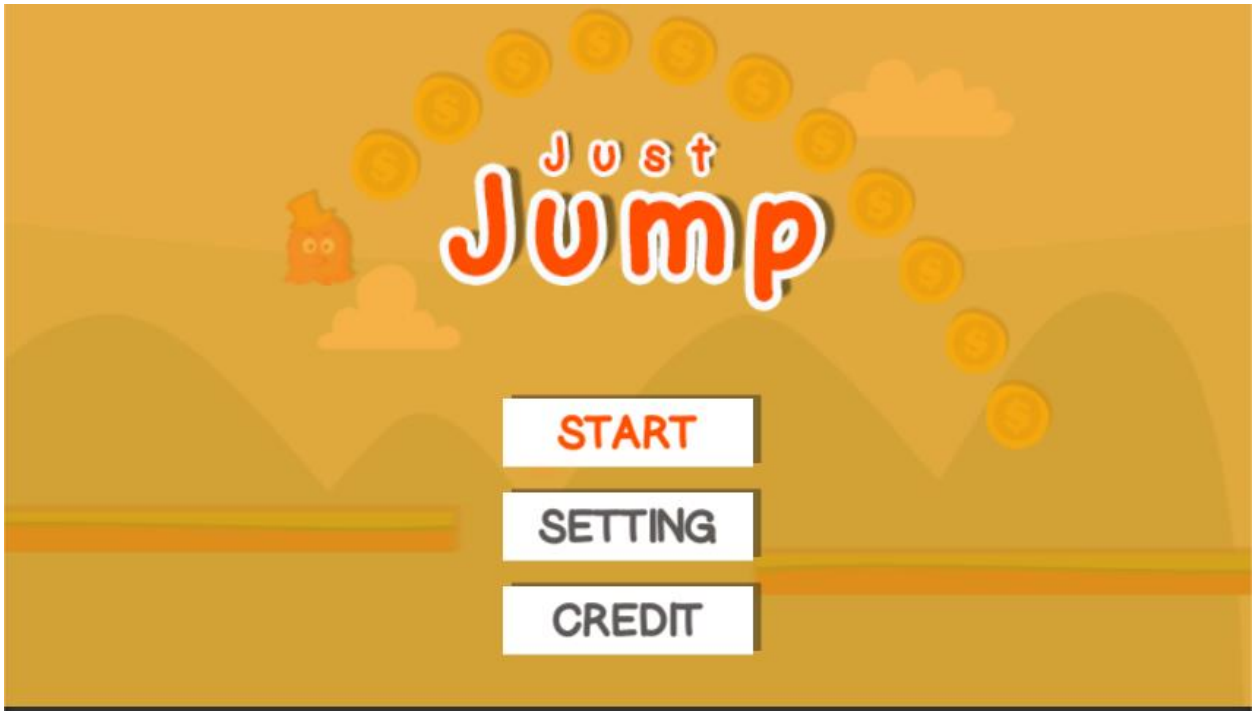
Tên	Kiểu	Mô tả
jumper image	Image	Image khi người chơi bị thua của game
replay button	Button	Dùng để quay lại GameScreen
back menu	Button	Dùng để đi đến StartScreen

3.2. Sơ đồ Use – case

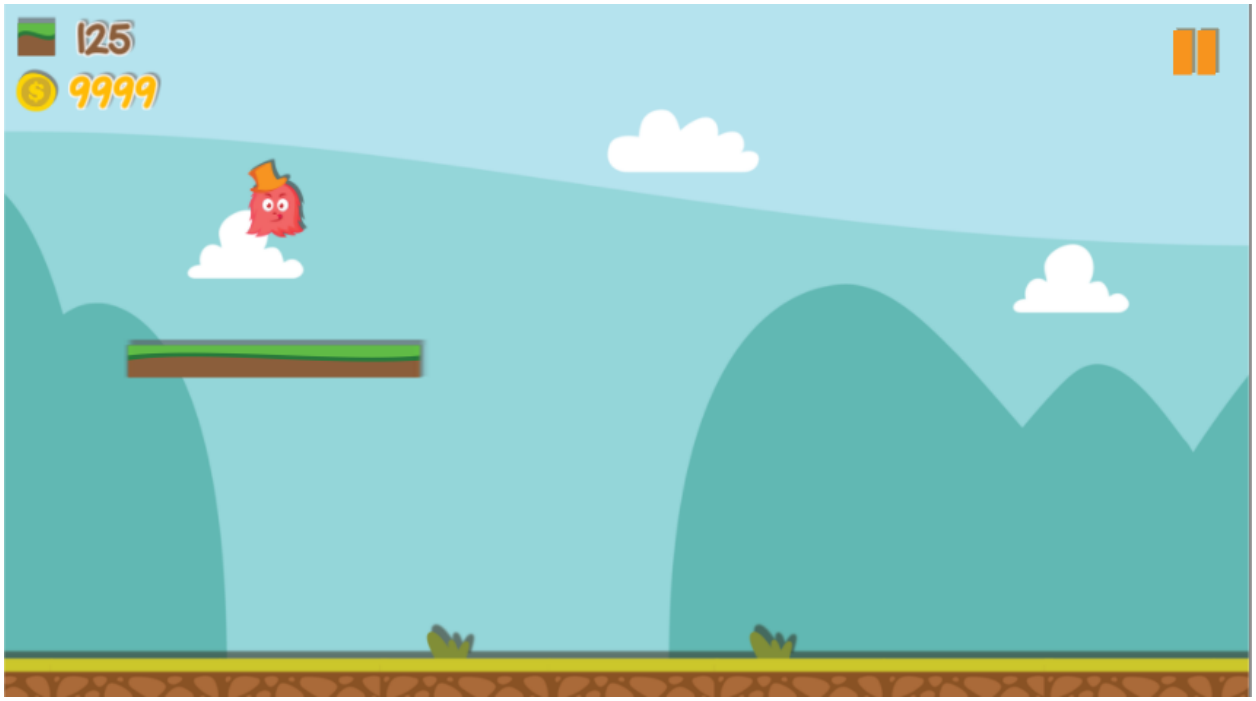


3.3. Giao diện ứng dụng

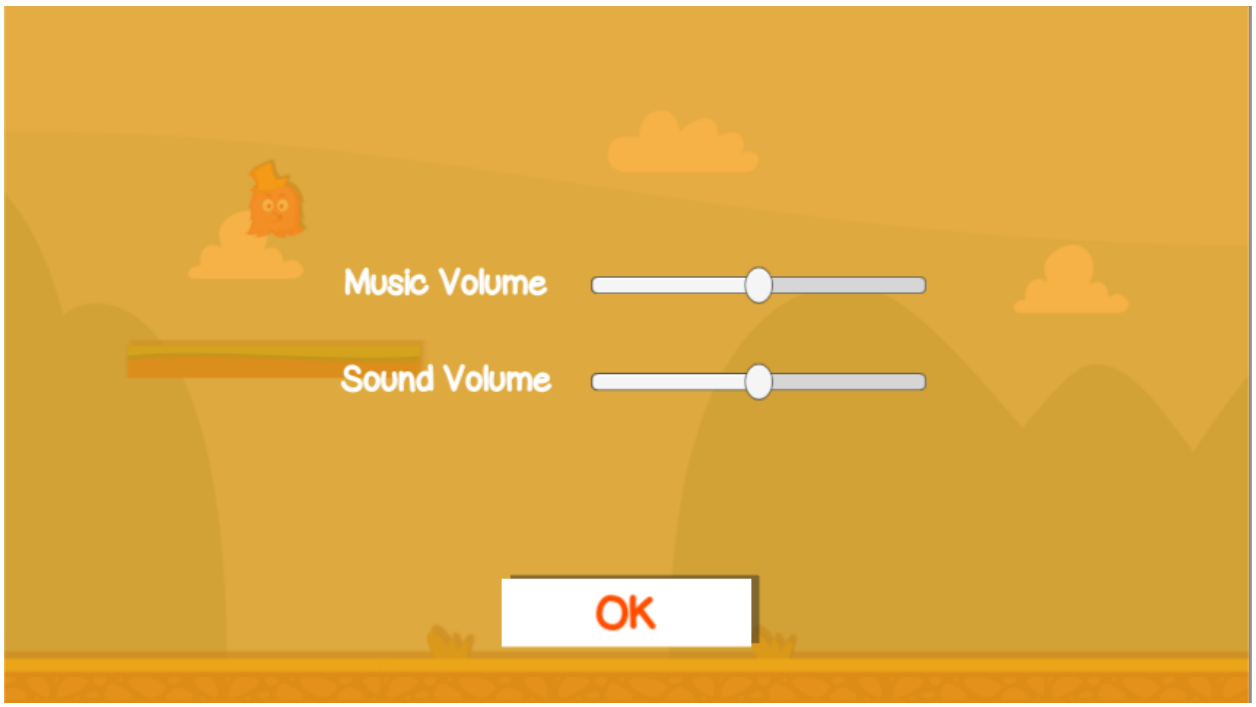
❖ **Giao diện StartScreen**



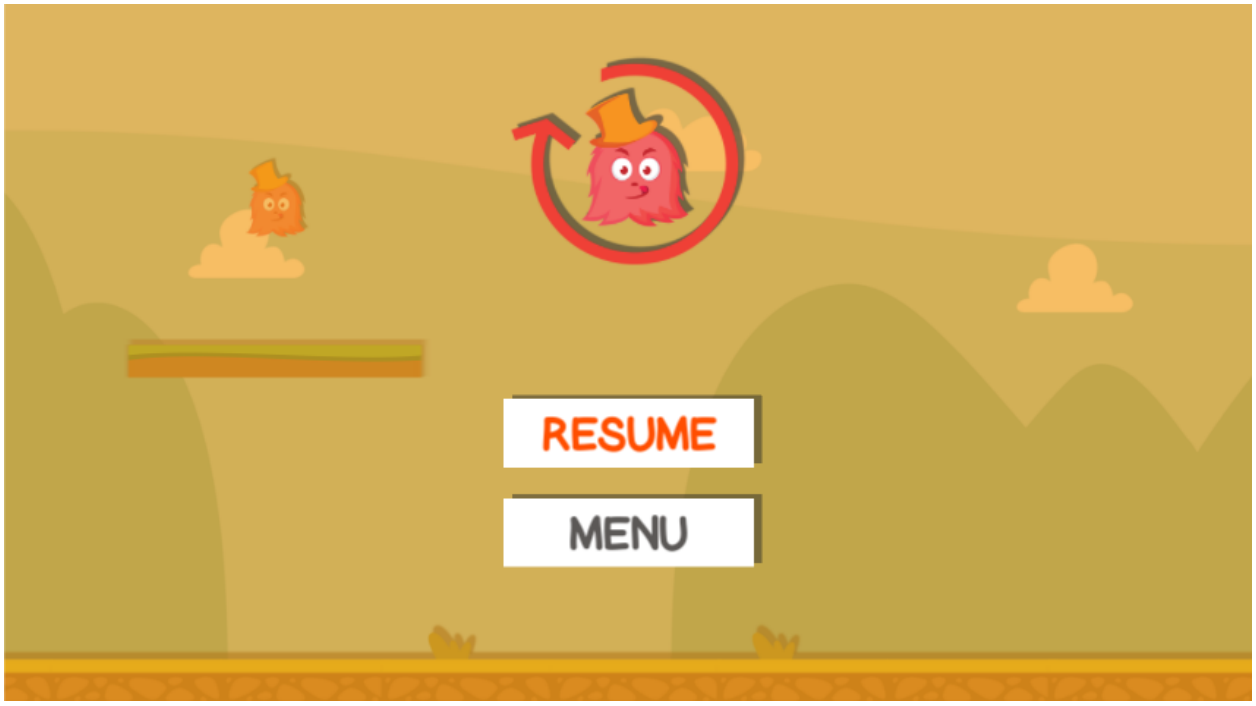
❖ **Giao diện GameScreen**



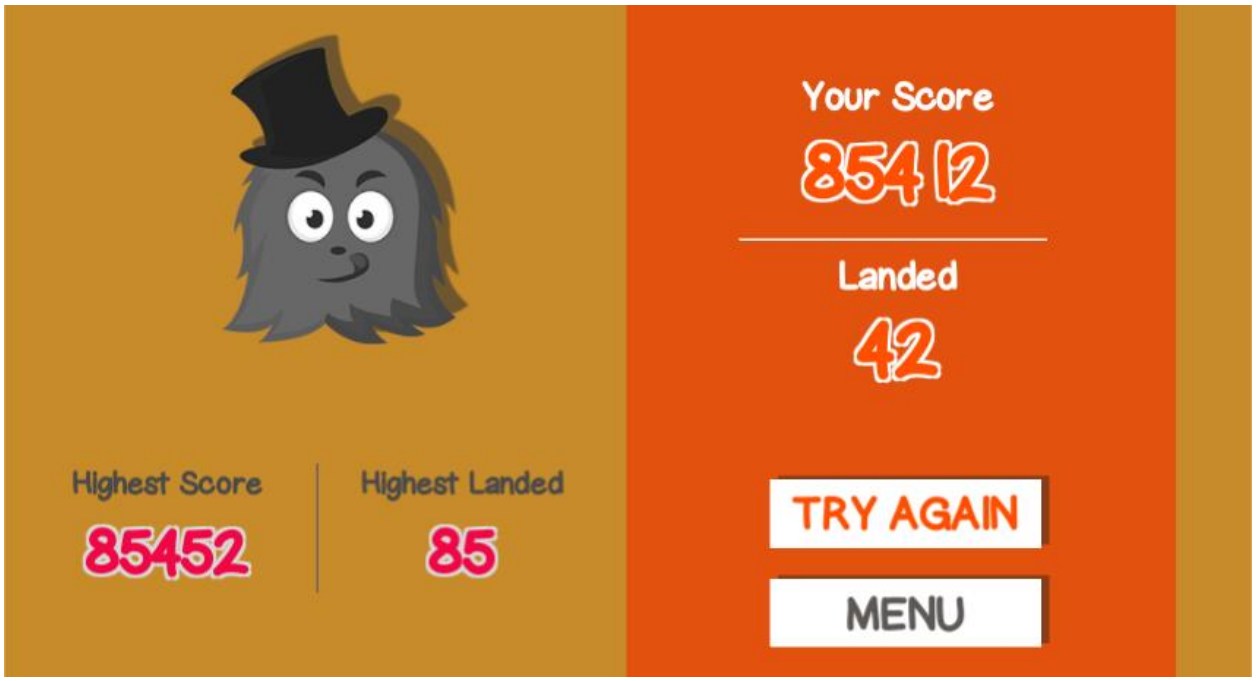
❖ **Giao diện SettingScreen**



❖ **Giao diện PauseScreen**



❖ **Giao diện ReplayScreen**



IV. PHỤ LỤC – HƯỚNG DẪN CÀI ĐẶT VÀ SỬ DỤNG

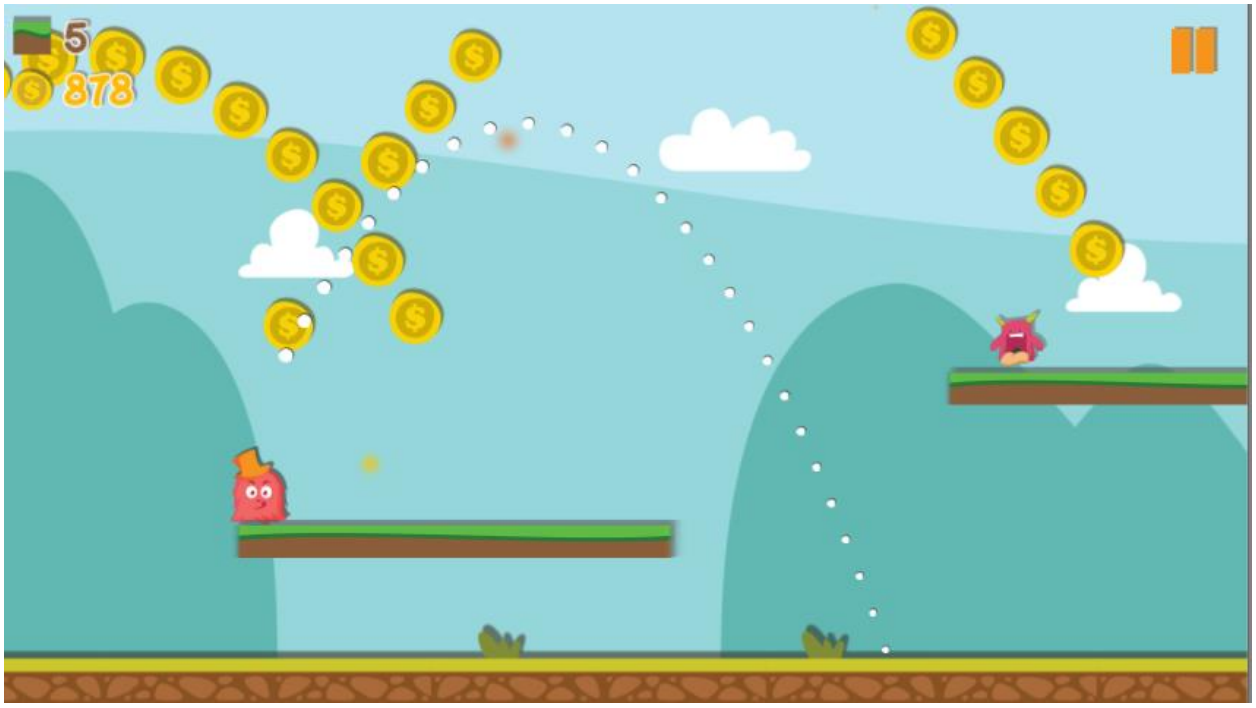
4.1. Hướng dẫn cài đặt

- Tải và chạy file justjump.apk
- Lưu ý, game chỉ chạy được trên hệ điều hành Android 2.2 trở lên.

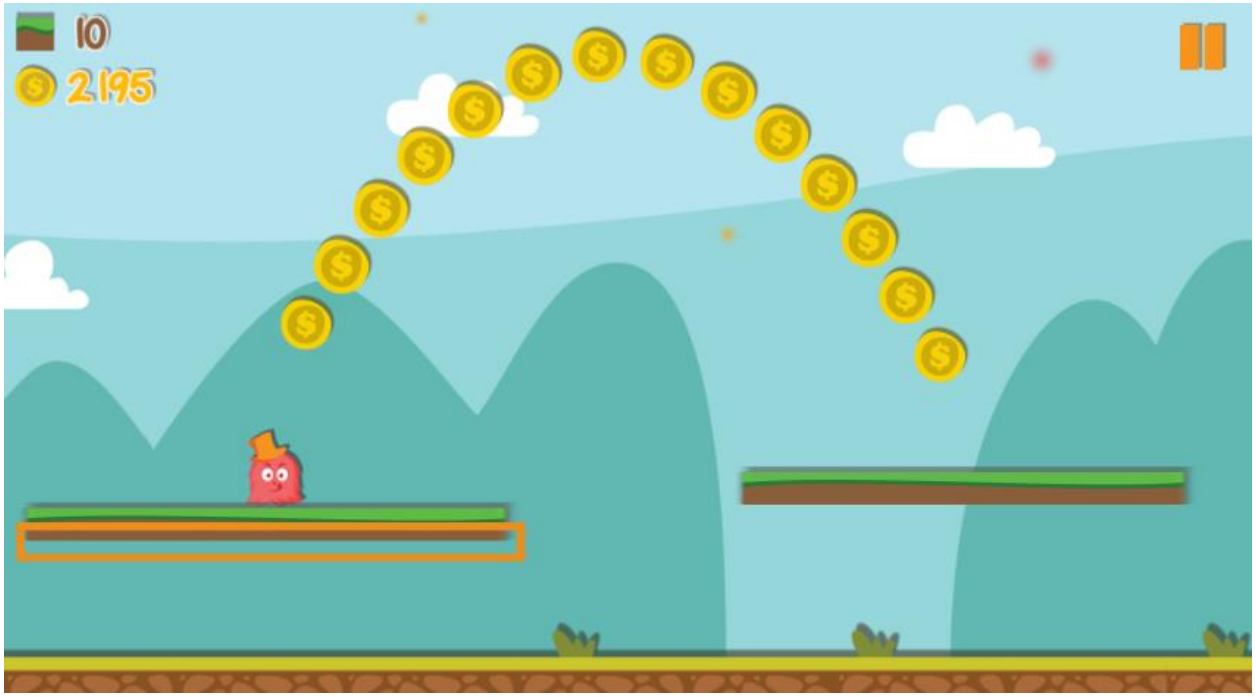
4.2. Sử dụng

- Khi màn hình game xuất hiện, nhấn START để bắt đầu chơi.
- Khi vào game, người chơi cần chờ cho nhân vật chạm vào land, lúc này người chơi cần giữ chạm màn hình để điều chỉnh lực nhảy của nhân vật.
- Khi đang nhảy, người chơi có thể chạm màn hình để kích hoạt chức năng bắn đạn.
- Nếu người chơi nhảy không chạm vào land, người chơi sẽ kết thúc lượt chơi và xuất hiện màn hình ReplayScene. Nếu muốn chơi lại, nhấn TRY AGAIN, nếu muốn về StartScreen, nhấn MENU.
- Trong quá trình đang chơi, nếu muốn dừng lại, người chơi có thể nhấn vào biểu tượng Pause ở góc trên phải màn hình.

- Có 2 loại trợ giúp:
  - Direction: Khi có sự trợ giúp này, sẽ xuất hiện quỹ đạo nhảy giúp người chơi có thể đạt Perfect khi thực hiện bước nhảy.



  - LineDirection: Khi có sự trợ giúp này, nếu land trùng khớp với viền màu cam, thì người chơi sẽ nhảy vào trung tâm của land tiếp theo.



- Có 3 loại Enemy: Một loại đứng yên trên land, một loại di chuyển trên land và một loại bay trên không.
- Nếu nhân vật chạm quái thì sẽ kết thúc lượt chơi.
- Nếu người chơi bắn hạ quái thì sẽ được cộng điểm, con nếu chỉ chạm land thì quái cũng chết nhưng người chơi không được cộng điểm.
- Độ khó tăng lên bằng cách land sẽ ngày càng ngắn lại nhưng vẫn có giới hạn nhỏ nhất.