

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG TRƯỜNG ĐẠI HỌC NHA TRANG
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN TỐT NGHIỆP

**THIẾT KẾ BÀI THỰC HÀNH ẢO ỨNG DỤNG TRONG
LĨNH VỰC SINH HỌC (MÔ PHỎNG ĐỊNH LƯỢNG
CHLOROPHYLL BẰNG PHƯƠNG PHÁP ĐO QUANG PHÔ)**

Giảng viên hướng dẫn: ThS. Đoàn Vũ Thịnh

Sinh viên thực hiện: Nguyễn Tấn Phát

Mã số sinh viên: 60136516

Khánh Hòa – 2022

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG TRƯỜNG ĐẠI HỌC NHA TRANG
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN TỐT NGHIỆP

**THIẾT KẾ BÀI THỰC HÀNH ẢO ỨNG DỤNG TRONG
LĨNH VỰC SINH HỌC (MÔ PHỎNG ĐỊNH LƯỢNG
CHLOROPHYLL BẰNG PHƯƠNG PHÁP ĐO QUANG PHÔ)**

GVHD: ThS. Đoàn Vũ Thịnh

SVTH: Nguyễn Tấn Phát

MSSV: 60136516

Khánh Hòa, Tháng 07/2022

LỜI CAM ĐOAN

Tôi xin cam đoan mọi kết quả của đề tài “**Thiết kế bài thực hành ảo ứng dụng trong lĩnh vực sinh học (Mô phỏng định lượng Chlorophyll bằng phương pháp đo quang phổ)**” là công trình nghiên cứu của cá nhân tôi và chưa từng được công bố trong bất cứ công trình khoa học nào khác cho tới thời điểm này.

Khánh Hòa, ngày 23 tháng 06 năm 2022

Tác giả luận văn

(kí và ghi rõ họ tên)

LỜI CẢM ƠN

Trong suốt thời gian thực hiện đề tài, tôi đã nhận được sự giúp đỡ của quý phòng ban Trường Đại học Nha Trang, Khoa Công nghệ Thông tin nói chung và Bộ môn Kỹ thuật phần mềm nói riêng đã tạo điều kiện cho tôi được hoàn thành đề tài. Đặc biệt là sự hướng dẫn tận tình của Ths. Đoàn Vũ Thịnh đã giúp tôi hoàn thành tốt đề tài.

Xin chân thành cảm ơn các quý thầy cô trong bộ môn Kỹ thuật phần mềm trong thời gian qua đã truyền đạt và trang bị kiến thức giúp em hoàn thành tốt đề tài tốt nghiệp. Cuối cùng tôi xin gửi lời cảm ơn chân thành đến gia đình và tất cả bạn bè đã giúp đỡ, động viên tôi trong suốt quá trình học tập và thực hiện đề tài.

Tôi xin chân thành cảm ơn.

Khánh Hòa, ngày 23 tháng 06 năm 2022

Tác giả luận văn

(kí và ghi rõ họ tên)

MỤC LỤC

LỜI CAM ĐOAN	i
LỜI CẢM ƠN.....	ii
DANH MỤC HÌNH	v
DANH MỤC BẢNG.....	viii
DANH MỤC CÁC KÝ HIỆU, TỪ VIẾT TẮT	ix
CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI.....	3
1.1 Nghiên cứu trong nước	3
1.2 Nghiên cứu ngoài nước	4
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT.....	7
2.1 Thí nghiệm ảo	7
2.2 Game Engine	7
2.2.1 Lựa chọn Game Engine.....	9
2.3 Unity Engine	10
2.3.1 Những thành phần cơ bản trong dự án Unity.....	11
2.3.2 Lập trình bằng giao diện kéo-thả	13
2.3.3 Lập trình dòng lệnh bằng Visual Studio	13
2.4 Microsoft Visual Studio và Ngôn ngữ lập trình C-Sharp	14
2.4.1 Microsoft Visual Studio	14
2.4.2 Ngôn ngữ lập trình C-Sharp (C#)	14
2.4.3 Sự kết nối giữa Visual Studio – C# - Unity	15
2.5 Blender	15
2.6 Adobe Illustrator	17
2.7 Adobe Photoshop	18
2.8 Trình chuyển đổi văn bản thành âm thanh Vbee.vn	18
CHƯƠNG 3. XÂY DỰNG CHƯƠNG TRÌNH MÔ PHỎNG	20
3.1 Chuẩn bị tổng quan cho môi trường làm việc.....	20
3.1.1 Unity Engine	20
3.1.2 Cài đặt Visual Studio và C# cho Unity	22

3.1.3 Blender	23
3.1.4 Adobe Illustrator	24
3.2 Xây dựng Scene	26
3.2.1 Scene 0 - Màn hình chính	26
3.2.2 Scene 1 – Đăng nhập.....	30
3.2.3 Scene 2 – Menu.....	33
3.2.4 Scene 3 – Giới thiệu	37
3.2.5 Scene 4 – Nguyên lí hoạt động của máy đo quang phổ	39
3.2.6 Scene 5 – Vật liệu và hóa chất	41
3.2.7 Scene 6 – Thực hành mô phỏng “Chuẩn bị vật liệu”	44
3.2.8 Scene 7 – Thực hành mô phỏng “ <i>Sử dụng máy đo quang phổ</i> ”	55
CHƯƠNG 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI.....	65
4.1 Kết luận	65
4.2 Hướng phát triển đề tài.....	66
TÀI LIỆU THAM KHẢO	68

DANH MỤC HÌNH

Hình 1.1. Giao diện mô phỏng “Bí quyết bảo quản thực phẩm” thực hiện bởi BASF ...	3
Hình 1.2. Chương trình mô phỏng “ <i>Chọn lọc tự nhiên</i> ”	4
Hình 1.3. Thí nghiệm ảo “Thời gian thực PCR”	5
Hình 1.4. Chương trình mô phỏng “ <i>Sử dụng Microscope</i> ”	5
Hình 1.5 Chương trình mô phỏng “ <i>Di truyền động vật</i> ”.....	6
Hình 2.1. Những Game Engine phổ biến nhất hiện nay.....	8
Hình 2.2. Logo của Unity	11
Hình 2.3. Giao diện đồ họa của Unity	12
Hình 2.4. Giao diện đồ họa của Animator Controller	13
Hình 2.5. Giao diện lựa chọn hàm cho GameObject.....	13
Hình 2.6. Hàm đã lập trình sẵn và C# Script Visual Studio.....	14
Hình 2.7. Giao diện đồ họa của Blender	15
Hình 2.8. Giao diện Modeling của Blender.....	16
Hình 2.9. Giao diện Animation của Blender	16
Hình 2.10. Giao diện Shading của Blender	17
Hình 2.11. Môi trường xây dựng đồ họa của Adobe Illustrator	17
Hình 2.12. Giao diện đồ họa của Adobe Photoshop	18
Hình 2.13. Danh sách giọng đọc và Giao diện chuyển đổi văn bản thành giọng nói ...	18
Hình 3.1. Giao diện của cây phả hệ GameHandler	20
Hình 3.2. Create Empty để tạo GameObject mới.....	20
Hình 3.3. Các thành phần đầu tiên của dự án Unity	21
Hình 3.4. Component Camera và Canvas	21
Hình 3.5. Tài nguyên Assets của chương trình mô phỏng	22
Hình 3.6. Giao diện cài đặt Visual Studio	23
Hình 3.7. Mở C# Script thông qua Unity	23
Hình 3.8. Môi trường làm việc của Blender.....	23
Hình 3.9. Các mục điều chỉnh khi Export	24
Hình 3.10. Các phần công cụ của Adobe Illustrator	25
Hình 3.11. Icon Button của Game và màn hình các thiết bị	25
Hình 3.12. Các thành phần của một background làm bằng Illustrator.....	26

Hình 3.13. Cây phả hệ và Giao diện đồ họa phân cảnh 0	27
Hình 3.14. Gán GameObject Button vào C# Script LeanTween	28
Hình 3.15. Cập nhật Scene bằng thư viện SceneManagement và chọn OnClick() để kích hoạt Scene	29
Hình 3.16. Thành phần của phân cảnh 1 và Canvas Renderer.....	30
Hình 3.17. Cây phả hệ và Giao diện phân cảnh “Đăng nhập”.....	30
Hình 3.18. Thông báo khi InputField sai thông tin và bỏ trống	31
Hình 3.19. Cài đặt hàm Animation và CheckValid()	32
Hình 3.20. Khai báo đối tượng lưu lại thông tin và GameObject cho C# Script	32
Hình 3.21. Cài đặt ResetField() cho Button và LeanTweener	33
Hình 3.22. Các thành phần của phân cảnh “Menu”.....	33
Hình 3.23 Gán văn bản vào phần nhập liệu công cụ chuyển đổi âm thanh Vbee.vn....	34
Hình 3.24 Bấm “Tải về” khi trạng thái đạt 100%	34
Hình 3.25. Cây phả hệ phân cảnh và Thông báo chào mừng	35
Hình 3.26. Gán âm thanh vào AudioClip và nút “Tắt”	36
Hình 3.27. Gán hàm chuyển Scene cho GameObject	37
Hình 3.28. Template “Giới Thiệu” và Tạo hiệu ứng đối tượng đồ họa.....	38
Hình 3.29. Cây phả hệ và khung hình đầu của phân cảnh 3	38
Hình 3.30. Gán thư viện Animation cho GameObject	38
Hình 3.31. Giao diện xử lí Camtasia	39
Hình 3.32. Cây phả hệ và Giao diện đồ họa của phân cảnh 4.....	39
Hình 3.33. Điều chỉnh thuộc tính Video Player cho GameObject	40
Hình 3.34. Gán hàm chuyển Scene cho Button.....	41
Hình 3.35. Các Components UI của Scene	41
Hình 3.36. Các hình ảnh vật liệu sau khi được xử lí trên Photoshop	42
Hình 3.37. Giao diện đồ họa của phân cảnh 5 và cây phả hệ.....	42
Hình 3.38. Template GameObject tiêu chuẩn	43
Hình 3.39. Cài đặt sự kiện OnClick() cho Button	43
Hình 3.40. Template cho vật liệu khi được nhấn vào	43
Hình 3.41. Khai báo các đối tượng vào C# Script.....	44
Hình 3.42. Giao diện xử lí model và Bảng thuộc tính 3D object.....	45
Hình 3.43. Sổ hướng dẫn trong Game và cây phả hệ So Tay Tutorial	45

Hình 3.44. Cài đặt OnClick() cho hai nút tương tác của SoTayTutorial	46
Hình 3.45. Giao diện thực hành “ <i>Cắt Lá Theo Miếng Mica</i> ” và cây phả hệ.....	46
Hình 3.46. Timeline Animation của đối tượng Kéo.....	47
Hình 3.47. Thông báo hoàn thành và Gán sự kiện OnClick().....	48
Hình 3.48. Giao diện tương tác của “ <i>Đo lá bằng máy cân phân tích</i> ” và <i>Giao diện màn hình của máy cân phân tích</i>	49
Hình 3.49. Khởi tạo dữ liệu animation cho máy cân phân tích.....	49
Hình 3.50. Khai báo đối tượng TextCountNumber và gán TextCountNumberUpdated vào Button tương tác	51
Hình 3.51. Giao diện đồ họa “ <i>Xử lý lá bằng hóa chất và chày cối</i> ”	52
Hình 3.52. Khai báo đối tượng Animator để kích hoạt Trigger	52
Hình 3.53. GameObject Micropette được giả lập như một Scene	53
Hình 3.54. Giao diện đồ họa của “ <i>Sử dụng máy ly tâm cho vật mẫu</i> ” và cây phả hệ ..	53
Hình 3.55. Hiệu chỉnh số vòng quay của máy ly tâm	54
Hình 3.56 Khai báo các đối tượng cần hiển thị ở màn hình máy ly tâm.....	55
Hình 3.57. Hình ảnh máy ly tâm hoạt động trong Unity.....	55
Hình 3.58. Mặt trước và mặt sau của máy đo quang phổ trong Blender	56
Hình 3.59. Giao diện “ <i>Sử dụng máy đo quang phổ</i> ” và cây phả hệ	56
Hình 3.60. Khai báo các đối tượng Animator vào C# script quản lý	58
Hình 3.61. Các trang màn hình của máy đo quang phổ	58
Hình 3.62. Khai báo đối tượng GameObject vào C# Script.....	59
Hình 3.63. Cài đặt C# Script cho các Button thực thi	59
Hình 3.64 Giao diện Số báo cáo và cây phả hệ.....	59
Hình 3.65. Thông báo lỗi và đối tượng Text truyền dữ liệu.....	62
Hình 3.66. Components của phân cảnh được thiết kế bằng Adobe Illustrator.....	63
Hình 3.67. Giao diện của Scene kiểm tra và cây phả hệ	63
Hình 3.68. Thông báo hoàn thành và chưa hoàn thành.....	64
Hình 3.69. Khai báo các đối tượng của số báo cáo để kích hoạt các sự kiện	64

DANH MỤC BẢNG

Bảng 2.1. Đặc điểm của những GE phô biến trên thị trường	10
Bảng 4.1. Bảng so sánh các tính năng của những dự án trong nước.....	65

DANH MỤC CÁC KÝ HIỆU, TỪ VIẾT TẮT

2D	Two Dimensions (Không gian 2 chiều)
3D	Three Dimensions (Không gian 3 chiều)
API	Application Programming Interface (Giao diện chương trình ứng dụng)
CSS	Cascading Style Sheets (Tập tin định kiểu theo tầng)
GE	Game Engine (Phần mềm sản xuất trò chơi điện tử)
GIF	Graphics Interchange Format (Định dạng Trao đổi Hình ảnh)
HEX	Hexadecimal (Hệ Thập lục phân)
HTML5	HyperText Markup Language 5 (Ngôn ngữ Đánh dấu Siêu văn bản)
IDE	Integrated Development Environment (Môi trường phát triển tích hợp)
JPG	Joint Photographic Experts Group (Tiêu chuẩn chung cho định dạng ảnh)
JS	JavaScript (Ngôn ngữ lập trình JavaScript)
MD	Mobile Device (Thiết bị di động)
PC	Personal Computer (Máy tính cá nhân)
PNG	Portable Network Graphics (Đồ họa mạng di động)
XR	Extended Reality (Thực tế mở rộng)
WebGL	Web Graphics Library (Thư viện đồ họa của web)

LỜI MỞ ĐẦU

Lợi ích của các ứng dụng khoa học-công nghệ mang lại đã được chứng tỏ thông qua nhiều lĩnh vực, ví dụ như ứng dụng Internet/4G để học, làm việc trực tuyến, ứng dụng IoT cho Smarthome hoặc du lịch khắp nơi thông qua hệ thống kính thực tế ảo... Sự hiệu quả đó được khẳng định một lần nữa vào việc ứng dụng trong Công nghệ Sinh học, xây dựng chương trình mô phỏng, thí nghiệm ảo bằng các phần mềm mô phỏng, hay còn được gọi là Game Engine.

Đa phần thế hệ sinh viên thời nay đều sẽ có các thiết bị điện tử như máy tính cá nhân hoặc điện thoại thông minh được kết nối Wifi/Internet. Bên cạnh đó, các trường đại học đều bố trí một hoặc nhiều phòng tin học có trang thiết bị kết nối Internet đầy đủ để hỗ trợ sinh viên trong việc học tập. Vì thế, mở ra nhiều hướng tiếp cận về cách học tập hơn. Sinh viên có thể học bằng các bài giảng trực tuyến (E-learning) trên hệ thống website của Nhà trường. Tương tự như các bài giảng trực tuyến, các thí nghiệm ảo, chương trình mô phỏng có thể được lập trình và đăng tải lên trang E-Learning của Nhà trường và để sinh viên thực hành trực tuyến. Bằng cách đó, sẽ giúp sinh viên có sự chuẩn bị tốt hơn khi thực hành thí nghiệm trực tiếp tại phòng thí nghiệm, hạn chế các trường hợp xấu xảy ra trong phòng thí nghiệm vì sơ suất như làm vỡ các dụng cụ thí nghiệm, các trang thiết bị đắt tiền.

Nhu cầu về luyện tập thực hành các thí nghiệm của sinh viên dễ dàng nhận thấy khi các thí nghiệm sinh học được thực hiện khá tốn kém và bị hạn chế số lượng người thực hiện. Ví dụ như bài thí nghiệm “*Định lượng Chlorophyll bằng phương pháp đo quang phổ*” được thực hiện tại Trường Đại học Nha Trang, chỉ riêng hai thiết bị chuyên dụng là máy đo quang phổ và máy ly tâm có giá trị rất lớn. Nhưng mỗi lần thí nghiệm chỉ giới hạn ở số lượng 4 sinh viên thực hiện. Cùng với đó, một số thí nghiệm đặc biệt chỉ dành cho những học viên, những bạn sinh viên sẽ chưa được tiếp cận đến. Để bù đắp vào nhu cầu đó của sinh viên, việc xây dựng các chương trình mô phỏng cho thí nghiệm là cần thiết cho tương lai.

Trong phạm vi nghiên cứu đề tài, chương trình mô phỏng sẽ được viết bằng ngôn ngữ C# và xây dựng trong môi trường Unity Engine. Chương trình sẽ bao gồm các cảnh/phân trang: màn hình chính, đăng nhập, menu, giới thiệu, vật liệu và hóa chất, thực hành và kiểm tra kiến thức.

Kết quả của chương trình mô phỏng bài thí nghiệm “*Định lượng Chlorophyll bằng phương pháp đo quang phổ*” đã đạt được những yêu cầu đề ra, với việc xây dựng chính xác các vật liệu, hóa chất bằng vật thể 3D, mô phỏng đủ các bước của kịch bản thí nghiệm, giao diện tương tác rõ ràng, dễ sử dụng. Từ đó, giúp cho sinh viên có thể dễ dàng tiếp cận, có sự chuẩn bị kĩ và kiến thức vững chắc cho bài thí nghiệm.

CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI

Tính tới thời điểm hiện tại, khái niệm về thí nghiệm ảo hay chương trình mô phỏng thí nghiệm đã được biết đến và sử dụng rộng rãi ở nhiều trường, nhiều cấp học từ Đại học đến Trung học cơ sở. Những nền tảng E-Learning cung cấp cho các lớp học mà giảng viên có thể dạy với các công cụ tương tác hành vi thay vì PowerPoint tĩnh như trước kia. Khái niệm học một chiều đang dần được thay thế bởi việc vừa học vừa tương tác. Phương pháp học này sẽ giúp người học tăng tính chủ động trong việc học tập, tạo sự hứng thú, kích thích quá trình học tập thay vì kiểu học bị động trước kia. Điều đó cho thấy rằng, việc ứng dụng các chương trình mô phỏng, thí nghiệm ảo vào giáo trình giảng dạy là xu hướng mới của nền giáo dục hiện nay. Tuy nhiên, đa phần các chương trình mô phỏng hiện nay chỉ được xây dựng bởi các tổ chức nước ngoài và chỉ một số ít trong đó có hỗ trợ ngôn ngữ Tiếng Việt. Điều này gián tiếp gây khó khăn cho người học tiếp cận các chương trình mô phỏng, thí nghiệm ảo. Từ đó cũng cho thấy rằng, các chương trình mô phỏng, thí nghiệm ảo vẫn chưa được chú trọng và đầu tư nhiều ở Việt Nam.

1.1 NGHIÊN CỨU TRONG NUỚC

Các thí nghiệm ảo, chương trình mô phỏng do người Việt còn ít và chưa được đầu tư kĩ lưỡng về mặt hình thức, nội dung và thường chỉ được làm trên nền tảng môi trường 2D, chưa được phổ cập rộng rãi. Bên cạnh đó, các chương trình mô phỏng có hỗ trợ Tiếng Việt còn khá ít. Một số ví dụ như bài thí nghiệm ảo về “*Bí quyết bảo quản thực phẩm*” (<https://thinghiemvui.bASF.com>) do BASF Kids’ Lab thực hiện.



Hình 1.1. Giao diện mô phỏng “*Bí quyết bảo quản thực phẩm*” thực hiện bởi BASF

Phòng thí nghiệm ảo BASF là một sáng kiến thuộc chương trình BASF Kids’ Lab (Bé làm thí nghiệm), một chương trình giáo dục hóa học thực hành dành cho học sinh tiểu học trên toàn thế giới. Đến nay, đã có 10 thí nghiệm ảo về khoa học đã được giới thiệu đến học sinh tiểu học Việt Nam. Bên cạnh thí nghiệm trên, học sinh có thể trải

nghiệm trên nền tảng 7 thí nghiệm khác là: Năng lượng mặt trời, hạt xốp bí ẩn, truy tìm dấu vết, bong bóng tinh nghịch, tiệm bánh kỳ diệu, giải cứu nước bẩn...



Hình 1.2. Chương trình mô phỏng “Chọn lọc tự nhiên” thực hiện bởi PhET

PhET Interactive Simulation (<https://phet.colorado.edu/>) là một dự án do nhóm sinh viên tại trường Đại học Colorado Boulder thực hiện, xây dựng các chương trình mô phỏng và được hỗ trợ hơn 65 ngôn ngữ khác nhau, trong đó có cả Tiếng Việt.

Chương trình mô phỏng “Chọn lọc tự nhiên” (Hình 1.2) được thực hiện bởi đội ngũ của PhET, sử dụng HTML5/JS/CSS. Mô phỏng lại quá trình chọn lọc tự nhiên của Charles Darwin dựa trên nền tảng đồ họa 2D. Chương trình mô phỏng cho phép người học điều chỉnh các nhân tố môi trường, tính đột biến qua các thẻ hệ và biểu diễn thông tin bằng biểu đồ.

Nhìn chung, các chương trình mô phỏng hiện nay ở trong nước chỉ hỗ trợ các chương trình ở cấp Tiểu Học, Trung học Cơ sở và Trung học Phổ thông. Các bài thí nghiệm sinh học ở trình độ Đại học vẫn chưa được xây dựng đủ để đáp ứng cho đối tượng sinh viên.

1.2 NGHIÊN CỨU NGOÀI NUỐC

Trên thế giới, đã có nhiều tổ chức xây dựng các chương trình mô phỏng dựa trên kịch bản thí nghiệm thực tế như NCBioNetWork (<https://www.ncbionetwork.org/>) hoặc Labster (<https://www.labster.com/>), Praxilabs (<https://praxilabs.com/>).... Các chương trình mô phỏng đa dạng về mặt kịch bản thí nghiệm cho tới giao diện tương tác có độ chính xác cao. Kể đến như: Thí nghiệm ảo “Thời gian thật PCR của Praxilabs” (Hình 1.3) là chương trình mô phỏng lại quy trình thực hiện thí nghiệm sinh học phân tử dựa trên phản ứng chuỗi polymerase (PCR). Nó theo dõi sự khuếch đại của một phân tử DNA được nhắm mục tiêu trong quá trình PCR (nghĩa là trong thời gian thực), chứ không phải ở phần cuối của nó như trong PCR thông thường. PCR thời gian thực có thể được sử dụng định lượng (PCR thời gian thực định lượng) và bán định lượng (nghĩa là trên/dưới một lượng phân tử DNA nhất định) (PCR thời gian thực bán định lượng).



Hình 1.3. Thí nghiệm ảo “Thời gian thực PCR” của Praxilabs

Praxilabs là một trong những công ty start-up đáng chú ý trong mảng xây dựng các nội dung số về chủ đề giáo dục, đội ngũ của Praxilabs gồm những lập trình viên và chuyên gia về lĩnh vực giáo dục, những người hiểu được tầm quan trọng của các thí nghiệm trong giáo dục khoa học. Praxilabs đã phát triển các sản phẩm giúp các phòng thí nghiệm khoa học ảo có thể truy cập, sử dụng được và giá cả phải chăng cho các tổ chức giáo dục và trường học.

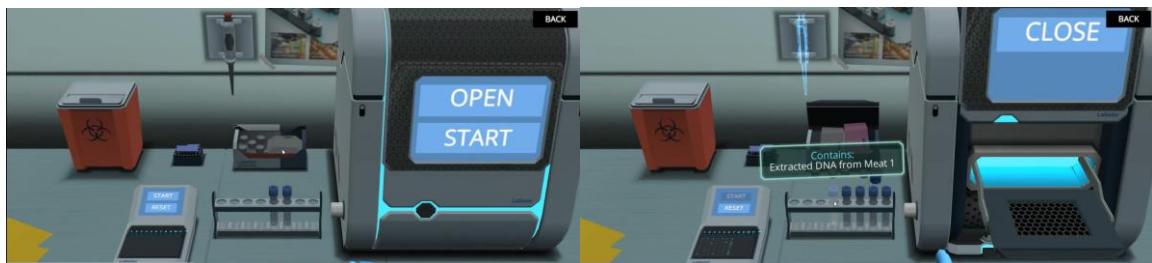


Hình 1.4. Chương trình mô phỏng “Sử dụng Microscope” của NCBioNetwork

Chương trình mô phỏng “Sử dụng Microscope” của NCBioNetwork (Hình 1.4) là một trong những chương trình mô phỏng nổi tiếng được thiết kế và lập trình bởi đội ngũ của NCBioNetwork. Chương trình mô phỏng lại cách hoạt động của máy hiển vi, kèm theo đó là các chỉ dẫn chi tiết và sách hướng dẫn cho những người mới. Chương trình mô phỏng này của NCBioNetwork giúp cho sinh viên tiếp cận được với máy hiển vi, hỗ trợ trong việc tìm hiểu và vận hành thiết bị này.

NCBioNetwork là tổ chức được thành lập vào năm 2004 thông qua sự tài trợ của Golden LEAF Foundation - một trong những tổ chức cung cấp các khóa học, hội thảo phát triển kỹ năng và các chương trình chứng chỉ về công nghệ sinh học, sản xuất dược phẩm và hóa chất. Đồng thời, còn là nơi xây dựng các chương trình mô phỏng thí nghiệm 3D chất lượng nhất trên thị trường hiện nay. Các thí nghiệm ảo trên có mức độ chính xác cao, thể hiện rõ ràng từng bước trong kịch bản của thí nghiệm cùng với số lần thực hành không giới hạn.

Bên cạnh đó, Labster (<https://www.labster.com/>) cũng là một trong những tổ chức nổi tiếng chuyên thiết kế và xây dựng các chương trình mô phỏng, môi trường thí nghiệm ảo hàng đầu thế giới. Các sản phẩm của Labster được đánh giá rất cao về chất lượng của chương trình mô phỏng với độ chính xác của các vật liệu 3D và độ thực tế của các phản ứng sinh - hóa học. Tuy nhiên, để sử dụng Labster người dùng cần bỏ ra một khoản kinh phí để thuê nền tảng này. Do đó, các bài giảng số và thí nghiệm ảo của Labster chỉ được truy cập miễn phí trong 30 ngày sau khi đăng ký. Những lần tiếp theo, để truy cập vào các nội dung này buộc phải trả một khoản phí tùy theo quy mô và số lượng người sử dụng.



Hình 1.5 Chương trình mô phỏng “Di truyền học động vật” thực hiện bởi Labster

Trong chương trình mô phỏng “Di truyền học động vật” của Labster (Hình 1.5), người học sẽ được tìm hiểu về sự di truyền Mendel và các đột biến trong chuỗi DNA có thể làm phát sinh kiểu hình. Bằng cách sử dụng phản ứng chuỗi polymerase (PCR) và điện di trên gel, người học sẽ thực hiện xác định kiểu gen nhằm xác định gen gây ra hiện tượng cơ bắp kép ở gia súc.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1 THÍ NGHIỆM ẢO

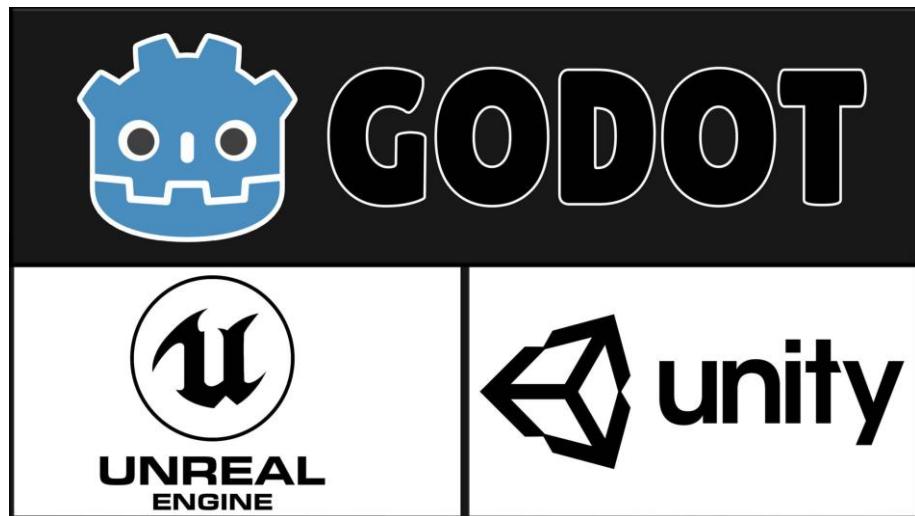
Thí nghiệm ảo là tập hợp các tài nguyên số đa phương tiện dưới hình thức đối tượng học tập, nhằm mục đích mô phỏng các hiện tượng vật lý, hóa học, sinh học...xảy ra trong tự nhiên hay trong phòng thí nghiệm, có đặc điểm là có tính năng tương tác cao, giao diện thân thiện với người sử dụng và có thể mô phỏng những quá trình, điều kiện tới hạn khó xảy ra trong tự nhiên hay khó thu được trong phòng thí nghiệm. Thí nghiệm ảo giúp giảm thiểu việc học, dạy chay thường gặp do thiếu phương tiện, điều kiện thí nghiệm giúp người học chủ động học tập phù hợp với tinh thần người học là trung tâm của giáo dục hiện đại.

Chương trình mô phỏng hay thí nghiệm ảo đã được ứng dụng nhiều trong hệ thống E-Learning của các trường học. Giảng viên có thể kết hợp bài học cùng với thí nghiệm ảo, giúp cho việc phổ cập kiến thức sinh động hơn. Cùng với đó, những chương trình mô phỏng thí nghiệm có thể lưu lại điểm số, thông tin của sinh viên, điều này giúp cho giảng viên có thể theo dõi tiến độ học tập của sinh viên chính xác hơn. Từ phía sinh viên có thể giữ được sự hứng thú đối với các bài giảng có kèm theo các chương trình mô phỏng, được tương tác với các đồ vật trong môi trường ảo, từ đó giúp các bạn sinh viên hình dung rõ hơn về những gì mình đang học và nắm vững những kiến thức đã học. Xét về khía cạnh chi phí, thực hành thí nghiệm trực tiếp sẽ tốn kém hơn so với việc thực hành trên thí nghiệm ảo. Thí nghiệm trực tiếp đòi hỏi sự chuẩn bị kĩ càng từ cả giảng viên và sinh viên về mặt kiến thức chuyên môn cũng như là chuẩn bị đầy đủ các vật liệu hóa chất cho bài thí nghiệm. Không như thí nghiệm trực tiếp, thực hành trên chương trình mô phỏng không yêu cầu bất kỳ công cụ và thiết bị phức tạp cụ thể nào để hoạt động. Tất cả những gì sinh viên cần là chương trình mô phỏng và một chiếc điện thoại thông minh hoặc một chiếc máy tính có kết nối Internet. Điều này, làm giảm chi phí và tài nguyên đáng kể cho cả từ phía sinh viên và Nhà trường.

2.2 GAME ENGINE

Game Engine (phần mềm sản xuất trò chơi điện tử) là một phần mềm được viết để thiết kế và phát triển trò chơi điện tử, hiểu đơn giản nó là loại phần mềm trung gian kết nối tương tác của nhiều ứng dụng trong cùng 1 hệ thống với nhau. Có rất nhiều loại Game Engine dùng để thiết kế trò chơi cho các hệ máy như hệ consoles(Máy chơi trò

chơi điện tử chuyên dụng) hay máy tính cá nhân. Chức năng cốt lõi của Game Engine phần lớn nằm trong tiếp xúc với phần cứng hay công cụ dựng hình (kết xuất đồ họa) cho các hình ảnh 2 chiều (2D) hay 3 chiều (3D), công cụ vật lý (hay công cụ tính toán và phát hiện va chạm), nhập/đọc tập tin chứa âm thanh và đồ họa, hát âm thanh, nâng cấp hình ảnh động (hoạt hình), trí tuệ nhân tạo cho những nhân vật trò chơi, phân luồng, tạo dòng dữ liệu xử lý, quản lý bộ nhớ, dựng ảnh đồ thị, và (nếu cần) kết nối mạng. Quá trình phát triển trò chơi tiết kiệm được rất nhiều thời gian và kinh phí vào việc tái sử dụng và tái thích ứng một phần mềm làm game để tạo nhiều trò chơi khác nhau.



Hình 2.1. Những Game Engine phổ biến nhất hiện nay

Game Engine cung cấp một bộ các công cụ phát triển trực quan và có thể tái sử dụng từng thành phần trong đó. Nói chung các bộ công cụ này cung cấp một môi trường phát triển tích hợp được đơn giản hóa. Phát triển ứng dụng nhanh cho trò chơi điện tử theo cách lập trình hướng dữ liệu. Những Game Engine này đôi khi còn được gọi là các “phần mềm trung gian cho trò chơi điện tử” (Game Middleware). Như ý nghĩa của thuật ngữ, chúng cung cấp một nền tảng phần mềm linh hoạt và dễ dàng sử dụng lại với mọi chức năng cốt lõi cần thiết ngay trong nó để có thể phát triển một trò chơi điện tử. Đồng thời giảm giá thành, độ phức tạp, và kịp thời hạn phát hành - tất cả các yếu tố quan trọng trong ngành công nghiệp sản xuất trò chơi điện tử đầy cạnh tranh.

Giống như các phần mềm trung gian khác, Game Engine thường cung cấp một nền tảng trừu tượng hóa, cho phép một trò chơi điện tử có thể chạy trên nhiều hệ máy bao gồm các hệ console (Máy chơi trò chơi điện tử chuyên dụng) hoặc máy tính cá nhân với một vài, nếu có, thay đổi trong mã nguồn của trò chơi điện tử đó. Thông thường, phần mềm trung gian cho trò chơi điện tử được thiết kế với một nền tảng kiến trúc dựa trên

các thành phần khác, cho phép các hệ thống khác nhau trong Game Engine có thể thay thế hoặc mở rộng với các phần mềm trung gian khác chuyên biệt hơn (và cũng đắt đỏ hơn) như là Havok cho hệ thống vật lý trong trò chơi điện tử, Miles Sound System cho âm thanh, hay Bink cho các đoạn video. Một số Game Engine như RenderWare thậm chí còn thiết kế như một loạt các kết nối lồng léo các phần mềm trung gian khác mà từ đó có thể chọn lọc và kết hợp để tạo ra một Game Engine theo yêu cầu, thay vì cách tiếp cận thông thường là mở rộng và tùy biến một cách linh hoạt để tạo ra giải pháp tích hợp. Tuy vậy, sự mở rộng vẫn cần thiết, đó vẫn là một ưu tiên cao do yêu cầu ứng dụng đa dạng trong Game Engine. Mặc dù có sự phức tạp trong tên gọi, các Game Engine vẫn thường được sử dụng cho các loại khác của ứng dụng tương tác với các yêu cầu đồ họa thời gian thực như giới thiệu các bản demo, dựng hình kiến trúc, đào tạo mô phỏng, và mô hình hóa môi trường, hoặc xây dựng thí nghiệm ảo.

2.2.1 Lựa chọn Game Engine

Trên thị trường hiện nay, có nhiều Game Engine tính phí, miễn phí hoặc có phiên bản dành cho Community, một số Game Engine phổ biến kể đến như:

- Godot Engine(<https://godotengine.org/>)
- Unreal Engine(<https://www.unrealengine.com/>)
- Unity Engine(<https://unity.com/>)

Mỗi Game Engine được xây dựng để phù hợp với những tiêu chí riêng của người sử dụng. Để chọn ra Game Engine phù hợp cho đề tài lần này, cần xem xét những tiêu chí phù hợp như ngôn ngữ sử dụng, mức độ hỗ trợ, đa nền tảng, dễ tiếp cận (*Bảng 2.1*).

Sau khi phân tích và nghiên cứu kỹ những tính năng của những Game Engine nổi tiếng trên thị trường hiện nay, tác giả đề tài quyết định chọn Unity Engine vì Game Engine này phù hợp với những tiêu chí đặt ra. Một điểm mạnh của Unity giúp Game Engine này vượt mặt qua các đối thủ của mình là cộng đồng người dùng của Unity. Với số lượng thành viên hơn 123.000 người, cộng đồng Unity phát triển mạnh mẽ giúp việc tự học và xây dựng chương trình trong Unity sẽ dễ dàng hơn khi khởi lượng chỉ dẫn, cũng như là chất lượng các bài hướng dẫn cho Unity nhiều đáng kể trên các diễn đàn và trên nền tảng trực tuyến như Youtube. Bên cạnh đó, tác giả đề tài đã có thời gian sử dụng ngôn ngữ lập trình C#. Vì thế, việc lập trình trong Unity sẽ thuận tiện hơn trong quá trình xây dựng chương trình mô phỏng. Và cuối cùng, đa phần các chương trình mô phỏng, thí nghiệm ảo của các tổ chức lớn như NCBioNetwork, Labster cũng xây dựng

bằng Unity Engine, điều đó cũng chứng tỏ phần nào sự ứng dụng rộng rãi của Unity vào các sản phẩm về bài giảng số, thí nghiệm ảo tương tác.

Bảng 2.1. Đặc điểm của những GE phổ biến trên thị trường

Tính năng	Unity Engine	Unreal Engine	Godot Engine
Ngôn ngữ	C#, UnityScript	C++	C++,C#, GDScript
Cộng đồng sử dụng	123.000+	42.600+	24.700+
Tính phí	Có phiên bản Community	Có phiên bản Community	Miễn phí
Nền tảng	PC,WEBGL, MD, XR,GC, AndroidTV, Nintendo 3DS,...	PC,WEBGL, MD, XR,GC	PC,WEBGL, MD, XR,GC
Đặc điểm nổi bật	<ul style="list-style-type: none"> ✓ Giao diện thân thiện ✓ Cộng đồng người dùng đông đảo ✓ Hệ thống AssetsStore đa dạng ✓ Phù hợp với dự án nhỏ 	<ul style="list-style-type: none"> ✓ Công nghệ đồ họa Next-Gen ✓ Đồ họa 3D chất lượng cao ✓ Phù hợp cho các dự án lớn 	<ul style="list-style-type: none"> ✓ Dễ tiếp cận ✓ Tài liệu tham khảo đa dạng

2.3 UNITY ENGINE

Unity là một Game Engine đa nền tảng được phát triển bởi Unity Technologies. Lần đầu tiên được công bố trên hệ điều hành OS X, tại Apple's Worldwide Developers Conference (2005), đã mở rộng cho nhiều nền tảng nhưng chủ yếu là phát triển trò chơi điện tử cho máy tính, máy trò chơi điện tử chuyên dụng (consoles) và điện thoại. Unity hỗ trợ đồ họa 2D và 3D, các chức năng được viết chủ yếu qua ngôn ngữ C#. UnityScript là một ngôn ngữ lập trình độc quyền có cú pháp tương tự JavaScript. Hỗ trợ đa dạng các APIs đồ họa trên các hệ điều hành thông dụng như: Direct3D (Windows, Xbox One); OpenGL (Linux, macOS, và Windows); OpenGL ES (Android, và iOS); WebGL (nền tảng web); và APIs độc quyền trên các máy chơi trò chơi điện tử (PS3, 4,5).



Hình 2.2. Logo của Unity

Trong các trò chơi đồ họa 2D, Unity cho phép nhập sprites (Hình ảnh tĩnh/động hai chiều khi được kết xuất vào môi trường Game Engine) và thế giới đồ họa 2D. Đối với nền tảng đồ họa 3D, Unity cho phép thiết lập các đập điểm kĩ thuật của các kết cấu và độ phân giải mà công cụ trò chơi hỗ trợ, cung cấp các hỗ trợ cho bump mapping (Một kỹ thuật lập bản đồ kết cấu trong đồ họa máy tính để mô phỏng các vết lồi và nếp nhăn trên bề mặt của một vật thể), reflection mapping (Một kỹ thuật chiếu sáng dựa trên hình ảnh hiệu quả để ước tính gần đúng sự xuất hiện của bề mặt phản chiếu bằng kết cấu được tính toán trước.), parallax mapping (Kỹ thuật ánh xạ vết sưng hoặc kỹ thuật ánh xạ thông thường), cảnh không gian ambient occlusion (SSAO), hiệu ứng bóng đổ bằng cách sử dụng shadow maps (Kỹ thuật đổ bóng đồ họa 3D), kết xuất thiết lập toàn cảnh đến hiệu ứng. Unity cũng cung cấp các dịch vụ cho nhà phát triển, bao gồm: Unity Ads, Unity Analytics, Unity Certification, Unity Cloud Build, Unity Everyplay, Unity API, Unity Multiplayer, Unity Performance Reporting and Unity Collaborate.

Unity nổi bật với khả năng xây dựng các trò chơi có thể thực thi trên nhiều nền tảng như: Android, Android TV, Facebook Gameroom, Fire OS, Gear VR, Google Cardboard, Google Daydream, HTC Vive, iOS, Linux, macOS, Microsoft HoloLens, Nintendo 3DS family, Nintendo Switch, Oculus Rift, PlayStation 4, PlayStation Vita, PlayStation VR, Samsung Smart TV, Tizen, tvOS, WebGL, Wii U, Windows, Windows Phone, Windows Store, và Xbox One.

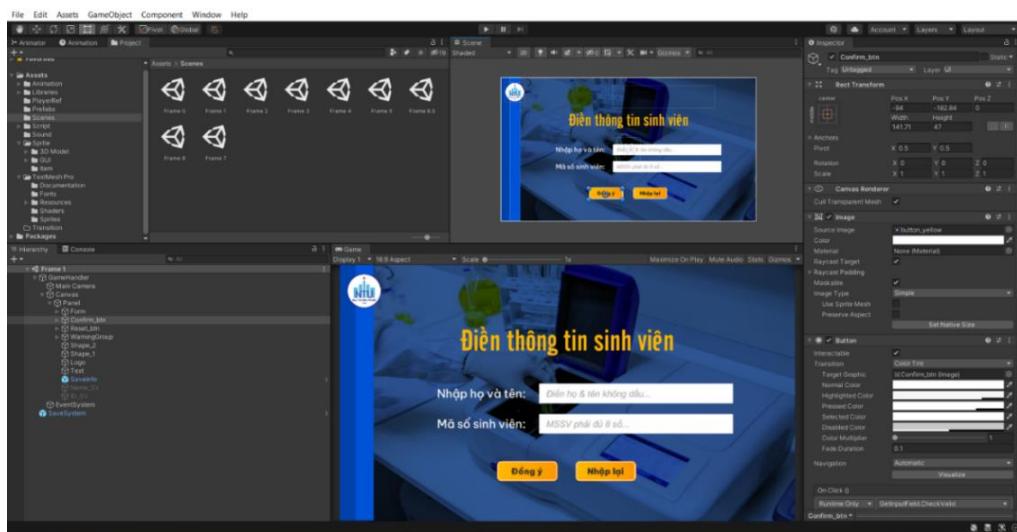
2.3.1 Những thành phần cơ bản trong dự án Unity

(1): Assets (Tài nguyên): Đây là mục quản lý tài nguyên của dự án, bao gồm: hình ảnh(hay còn được gọi là sprite 2D), model 3D, Animation controller, UnityScript, C# Script, Scene... Mọi thứ liên quan đến dự án đều sẽ được lưu trữ trong khu vực này.

(2): Scene (phân cảnh): Một dự án có một hoặc nhiều Scene, những trò chơi điện tử hạng AAA được thiết kế có quy mô lớn có thể chứa tới hàng trăm phân cảnh, ví dụ như: Ori and the Will of the Wisps, Hollow Knight,... Scene là đối tượng dùng để quản lý các GameObject thông qua cây phả hệ(Hierarchy). Mỗi Scene sẽ có cây phả hệ riêng

và chỉ có duy nhất một cây phả hệ cho một phân cảnh.

(3): Cây phả hệ (Hierarchy): Là khu vực dùng để hiển thị và quản lý các GameObject của Scene. Đây là vùng hiển thị trực quan giữa GameObject với Viewport đâu ra. Cây phả hệ giúp người dùng quản lý chặt chẽ các mối quan hệ giữa những GameObject với nhau.



Hình 2.3. Giao diện đồ họa của Unity

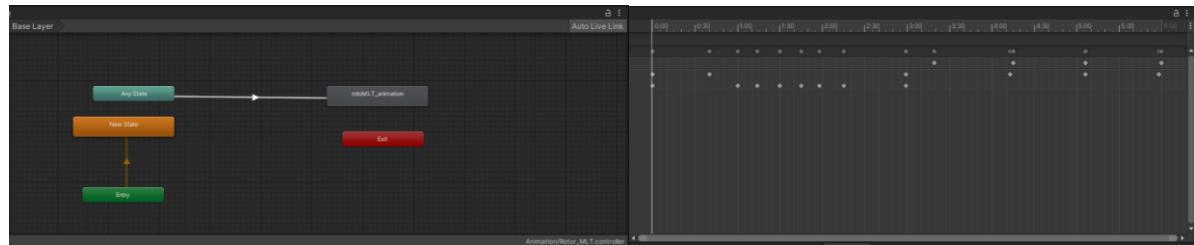
(4): GameObject (Đối tượng đồ họa): Là đối tượng đồ họa có thể hiện thị, hoặc không hiện thị trên Viewport Game. GameObject là thành phần cốt lõi để xây dựng bất kỳ dự án Unity nào. GameObject có thuộc tính được gọi là Component. GameObject được chia thành rất nhiều loại. Mỗi loại sẽ có thuộc tính và mục đích phục vụ riêng. Trong đồ án lần này, các GameObject thường được sử dụng nhiều nhất là UI, 3D/Object.

(5): Viewport (Vùng hiển thị đồ họa): Khu vực hiển thị các thành phần đồ họa và được chia ra thành hai loại chính là Viewport Scene và Viewport Game. Viewport Scene là vùng hiển thị toàn bộ các thành phần và thuộc tính của GameObject. Viewport Game là vùng hiển thị trình diễn giao diện của trò chơi sau khi được sản xuất.

(6): Component (Thành phần của GameObject) : Một GameObject thường được cấu tạo từ nhiều Component, có thể là hình ảnh, hành động của nhân vật, mã điều khiển, thư viện... Functions thường kế thừa từ MonoBehaviour class và có thể ghi đè bên trong những class còn có thể thực hiện cho những sự kiện quan trọng nào đó. Component sẽ dùng để gắn các thuộc tính mặc định hoặc custom script để phục vụ cho một mục đích cụ thể nào đó.

(7): Animation/Animator và Keyframe (Trình điều khiển hoạt họa) (*Hình 2.4*): Animation/Animator là bộ công cụ dùng để tạo hoạt ảnh cho các model 3D, cài đặt kịch

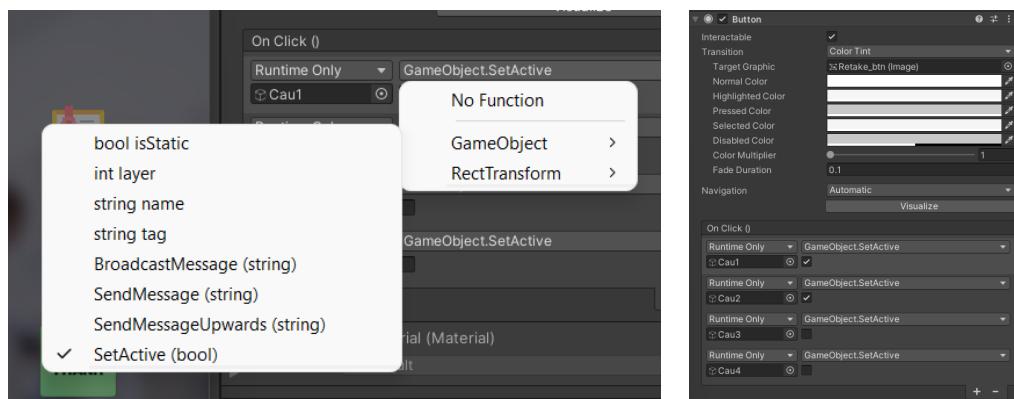
bản chuyển động cho các GameObject... Mỗi animation sẽ quản lý các thuộc tính của một hoặc nhiều GameObject. Từ đó cài đặt chuyển động, hoặc sự thay đổi thuộc tính theo thời gian bằng Keyframe - cột mốc thuộc tính thay đổi theo thời gian thực.



Hình 2.4. Giao diện đồ họa của Animator Controller

2.3.2 Lập trình bằng giao diện kéo-thả

Unity hỗ trợ người dùng lập trình các sự kiện thông qua giao diện kéo-thả làm cho việc lập trình trở nên tinh giản và tiết kiệm thời gian.



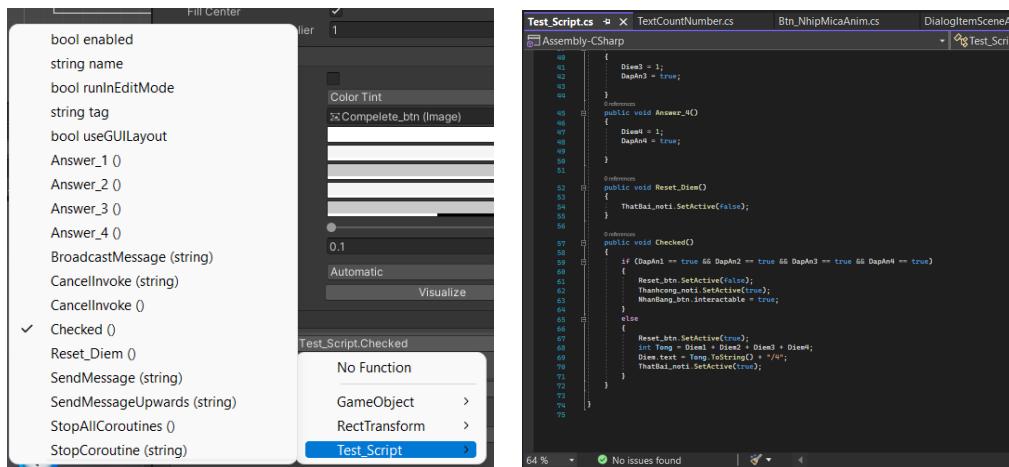
Hình 2.5. Giao diện lựa chọn hàm cho GameObject

Trong GameObject Button (*Hình 2.6*), Unity khởi động các hàm có sẵn như biến Boolean cho SetActive hoặc tạo biến string, int cho GameObject. Bên cạnh đó, lập trình viên có thể soạn thảo các đoạn mã với ngôn ngữ C# Script thêm vào Components GameObject và kéo-thả vào GameObject đã chọn.

2.3.3 Lập trình dòng lệnh bằng Visual Studio

Bên cạnh lập trình sự kiện bằng giao diện kéo-thả, người dùng có thể tự tạo những hàm tùy biến thông qua ngôn ngữ lập trình bằng ngôn ngữ C# để thiết lập những yêu cầu xử lý có độ phức tạp cao và quản lý nhiều sự kiện. Trình soạn thảo (IDE) phổ biến nhất là Visual Studio 2022. Unity tương thích cao với C# giúp cho người sử dụng có thể can thiệp sâu vào lõi của trò chơi được xây dựng bằng Unity.

Sau khi tạo và lập trình hàm Checked() trong C# Script “Test_Script”, bằng cách gán tệp C# vừa tạo vào phần Components của GameObject và cài đặt thông qua giao diện (*Hình 2.6*).



Hình 2.6. Hàm đã lập trình sẵn (Bên phải) và C# Script trong Visual Studio (Bên trái)

2.4 MICROSOFT VISUAL STUDIO VÀ NGÔN NGỮ LẬP TRÌNH C-SHARP

2.4.1 Microsoft Visual Studio (Phiên bản Community 2022)

Microsoft Visual Studio là một môi trường phát triển tích hợp (IDE) từ Microsoft được sử dụng để phát triển chương trình máy tính cho hệ điều hành Microsoft Windows, cũng như các trang web, các ứng dụng web và các dịch vụ web. Visual Studio sử dụng nền tảng phát triển phần mềm của Microsoft như Windows API, Windows Forms, Windows Presentation Foundation, Windows Store và Microsoft Silverlight.

Visual Studio bao gồm một trình soạn thảo mã hỗ trợ IntelliSense cũng như cài tiến mã nguồn. Trình gõ lỗi tích hợp hoạt động cả về trình gõ lỗi mức độ mã nguồn và gõ lỗi mức độ máy. Công cụ tích hợp khác bao gồm một mẫu thiết kế các hình thức xây dựng giao diện ứng dụng, thiết kế web, thiết kế lớp và thiết kế giản đồ cơ sở dữ liệu. Phần mềm này chấp nhận các plug-in nâng cao các chức năng ở hầu hết các cấp bao gồm thêm hỗ trợ cho các hệ thống quản lý phiên bản (Subversion) và bổ sung thêm bộ công cụ mới như biên tập và thiết kế trực quan cho các miền ngôn ngữ cụ thể hoặc bộ công cụ dành cho các khía cạnh khác trong quy trình phát triển phần mềm.

2.4.2 Ngôn ngữ lập trình C-Sharp (C#)

C-Sharp (còn gọi là C thăng) là một ngôn ngữ lập trình hướng đối tượng đa năng được phát triển bởi Microsoft. Tên của ngôn ngữ chỉ bao gồm ký tự thăng (#) theo Microsoft nhưng theo ECMA là C#, chỉ bao gồm dấu số thường. Microsoft phát triển C# dựa trên C++ và Java. C# được miêu tả là ngôn ngữ có được sự cân bằng giữa C++, Visual Basic, Delphi và Java.

C# là ngôn ngữ lập trình phản ánh trực tiếp đến .NET Framework. Mọi dữ liệu cơ sở đều là đối tượng, được cấp phát và hủy bỏ bởi trình dọn rác Garbage-Collector (GC),

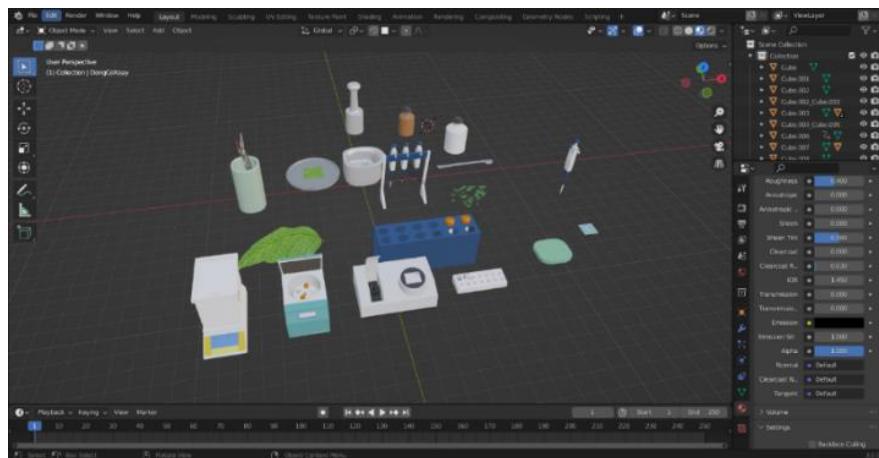
và nhiều kiểu dữ liệu trừu tượng như class, delegate, interface, exception... phản ánh rõ ràng những đặc trưng của .NET runtime. Bên cạnh đó, C# là ngôn ngữ được các lập trình viên trò chơi điện tử ưa chuộng. Vì gần như các Game Engine phổ biến hiện nay đều hỗ trợ ngôn ngữ lập trình C#.

2.4.3 Sự kết nối giữa Visual Studio – C# - Unity

IDE được ứng dụng trên Unity mặc định là Visual Studio và sử dụng ngôn ngữ lập trình C#. Visual Studio quản lý được các tệp C# Script rất hiệu quả trong quá trình xây dựng dự án. Một dự án Unity được lập trình đa đối tượng, có thể hàng chục hoặc hàng trăm đối tượng tùy vào quy mô của dự án.Thêm vào đó, Debug của Visual Studio có thể sử dụng để kích hoạt Play cho dự án của Unity, thao tác này sẽ đính kèm Debug của Visual Studio và Play của Unity giúp người dùng không cần phải chuyển cửa sổ biên dịch IDE để thực thi Unity, do đó tăng tốc quy trình gỡ lỗi của người dùng.

2.5 BLENDER (PHIÊN BẢN 3.0.0)

Blender (*Hình 2.7*) là một phần mềm đồ họa 3D miễn phí và nguồn mở, được sử dụng để sản xuất phim hoạt hình, kỹ xảo, ảnh nghệ thuật, mẫu in 3D, phần mềm tương tác 3D và trò chơi điện tử.



Hình 2.7. Giao diện đồ họa của Blender

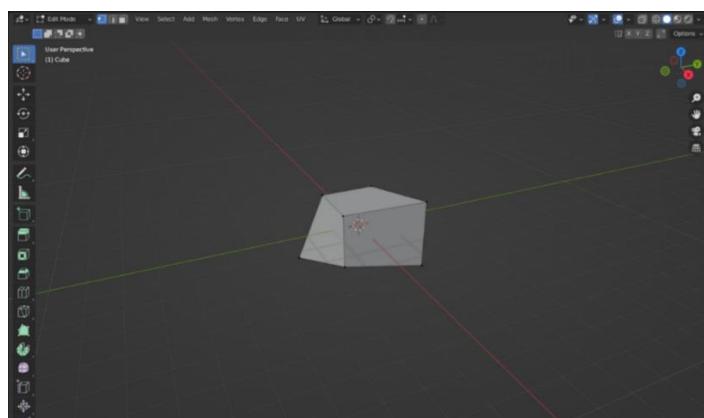
Các tính năng của Blender bao gồm tạo mẫu Model 3D, UV unwrapping (quá trình lập mô hình 3D chiếu hình ảnh 2D lên bề mặt của mô hình 3D để lập bản đồ kết cấu), áp vận bề mặt, mô phỏng khói, chất lỏng, hạt và chuyển động cơ thể, điêu khắc, hoạt họa, phối hợp chuyển động, camera tracking (Camera theo dõi đối tượng), Rendering (Kết xuất đồ họa) và chỉnh sửa video. Những tính năng nổi bật của Blender như:

- Tạo ra bộ khung hình một cách nhanh chóng
- Hỗ trợ giao diện cho các đoạn mã Python

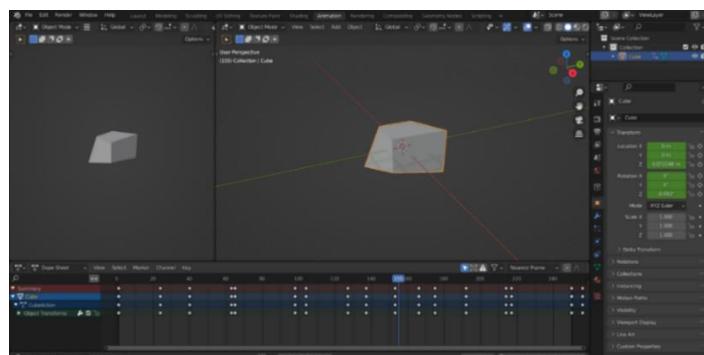
- Hỗ trợ giao diện tiếng Việt (từ ver 2.7.6)
- Hỗ trợ OSL (Ngôn ngữ tô bóng mở) của Sony

Công cụ của Blender:

(1): Modeling: Là một trong những công cụ cốt lõi và quan trọng của Blender. Công cụ này dùng để xây dựng, cấu tạo nên hình dạng cơ bản của vật thể 3D. Và đây cũng là một trong những công cụ tôi sử dụng nhiều nhất trong Blender. Modeling được chia ra thành rất nhiều thể loại khác nhau: Low Poly, Hard-Surface... Tùy vào mục đích và kỹ năng sử dụng, định hướng theo nhiều loại khác nhau.



Hình 2.8. Giao diện Modeling của Blender



Hình 2.9. Giao diện Animation của Blender

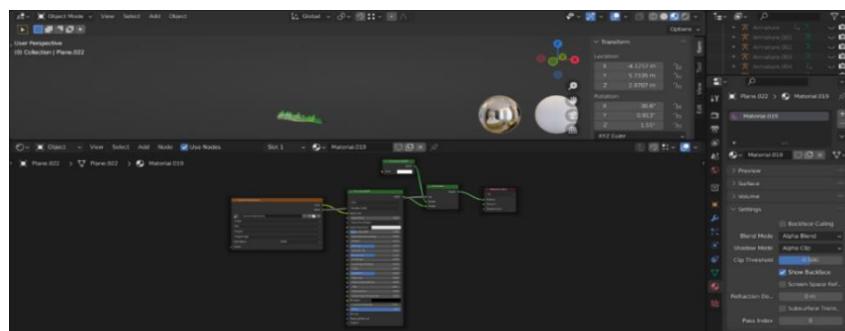
(1): Animation: Dùng để tạo chuyển động cho vật thể, tương tự như Unity. Tuy nhiên, chúng còn mạnh mẽ và quản lý hiệu quả hơn cả Unity. Dữ liệu animation của Blender có thể được lưu lại và trích xuất sử dụng cho Unity. Khi import vào Unity, dữ liệu animation của Blender có thể tự quy đổi sang dữ liệu chuyển động trong môi trường của Unity và chạy(Play) chuyển động như môi trường Blender.

Chú thích:

- Timeline của Animation: Được điều chỉnh bằng keyframe (biểu tượng hình chọn). Mỗi hình chọn sẽ tượng trưng cho một thuộc tính hay nhiều thuộc tính của vật thể biến chuyển theo thời gian thực.

➤ Bảng thuộc tính vật thể: Mỗi vật thể sẽ có bảng thuộc tính cơ bản dùng để điều chỉnh các thông số như chiều cao, chiều dài, độ xoay, độ cao, độ sâu,... Từng thông số trong bảng thuộc tính có thể thay đổi theo thời gian thực thông qua các thông số của keyframe.

(2): Shading: Là công cụ dùng để tạo node và trộn node mang các thuộc tính nhất định để tạo thành Material Output (Chất liệu đầu ra) cuối cùng. Chúng dùng để tạo vẻ ngoài của vật thể, mang lại các hiệu ứng và thuộc tính cho vật thể - bên cạnh đó chúng có thể tái sử dụng cho các vật thể khác.



Hình 2.10. Giao diện Shading của Blender

2.6 ADOBE ILLUSTRATOR

Adobe Illustrator là phần mềm thiết kế đồ họa chuyên về đồ họa vector. Sử dụng các thuật toán, các đối tượng hình học, text (đối tượng dạng chữ) kết hợp. Việc kết hợp giữa các hình dạng cơ bản đó với nhau để tạo thành một đối tượng vector. Illustrator được sử dụng chủ yếu để vẽ minh họa, hỗ trợ tạo nhân vật và phối cảnh phim hoạt hình. Adobe Illustrator sử dụng các thuật toán (vector) để tạo ra các đối tượng khác nhau. Điều này giúp cho sản phẩm của bạn có thể được lưu trữ và in ra ở mọi kích thước mà không hề ảnh hưởng đến chất lượng của sản phẩm cuối cùng. Không những thế, Illustrator còn được sử dụng để thiết kế UI (User Interface – Giao diện người dùng) cho ứng dụng và các phần mềm nói chung.



Hình 2.11. Môi trường xây dựng đồ họa của Adobe Illustrator

Adobe Illustrator sở hữu đặc điểm Artboard (Bảng vẽ đồ họa) tự do, giúp cho việc

quản lý các thành phần vector trở nên dễ dàng hơn. Đôi với các dự án sử dụng nhiều thành phần vector 2D để thiết kế UI, Adobe Illustrator hỗ trợ tốt cho các công cụ vẽ Vector Shape 2D – thành phần mà bất kỳ UI nào cũng buộc phải có.

2.7 ADOBE PHOTOSHOP

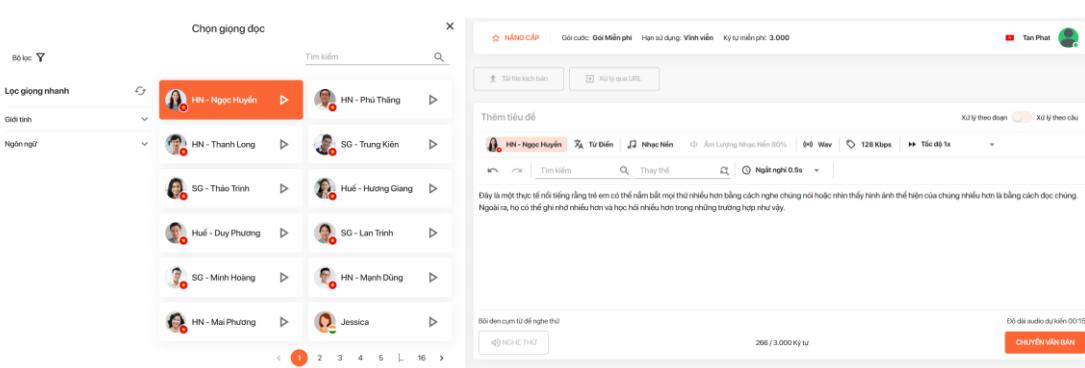
Adobe Photoshop là phần mềm đồ họa dùng để tạo hiệu ứng, chỉnh sửa nâng cao cho các tệp hình ảnh. Đây là phần mềm được sử dụng rộng rãi nhất trên thế giới vì tính đa dạng, công phu của các công cụ, thuật toán xử lý trong phần mềm. Adobe Photoshop chuyên dùng để xử lý các định dạng hình ảnh như PNG, JPEG, JPG, GIF... phục vụ cho các mục đích đồ họa khác nhau.



Hình 2.12. Giao diện đồ họa của Adobe Photoshop

Adobe Photoshop được sử dụng trong đề tài lần này với các mục đích như tái tạo ảnh nền cho phòng thí nghiệm. Tạo các lớp viền, nền và lớp đổ bóng cho các vật liệu, hóa chất, trang thiết bị của thí nghiệm ảo để tăng tính thu hút cho các đối tượng hình ảnh. Bên cạnh đó, Adobe Photoshop được sử dụng để tạo các vật phẩm đặc thù phục vụ cho thí nghiệm “*Dịnh lượng Chlorophyll bằng phương pháp đo quang phổ*”.

2.8 TRÌNH CHUYỂN ĐỔI VĂN BẢN THÀNH ÂM THANH VBEE.VN



Hình 2.13. Danh sách giọng đọc (Bên phải) và Giao diện chuyển đổi văn bản thành giọng nói (Bên trái)

Vbee.vn là nền tảng chuyển đổi văn bản sang âm thanh bằng công nghệ Học sâu Deep Learning và Trí tuệ Nhân tạo AI. Vbee.vn có hơn 190 giọng đọc khác nhau, trong đó có 16 giọng đọc Tiếng Việt chất lượng cao với nhiều âm giọng vùng miền (Hình 2.13). Nền tảng này được ứng dụng trong chương trình mô phỏng “*Định lượng Chlorophyll bằng phương pháp đo quang phổ*” để chuyển các đoạn văn bản hướng dẫn thành giọng nói hướng dẫn giúp cho người học thích thú hơn trong quá trình sử dụng.

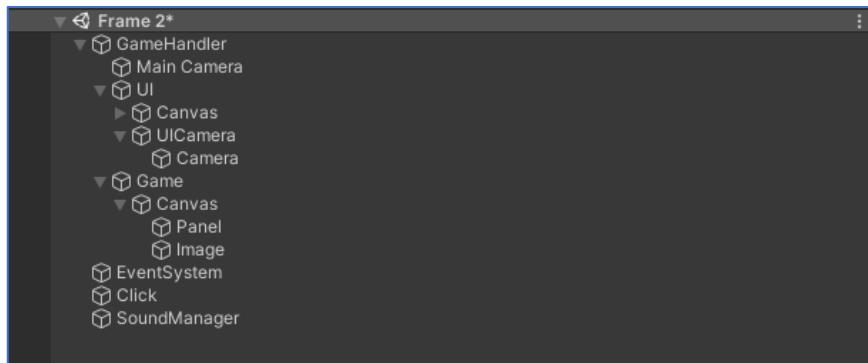
CHƯƠNG 3. XÂY DỰNG CHƯƠNG TRÌNH MÔ PHỎNG

3.1 CHUẨN BỊ TỔNG QUAN CHO MÔI TRƯỜNG LÀM VIỆC

3.1.1 Unity Engine

Cài đặt GameHandler

GameHandler và Canvas đều là các GameObject với thuộc tính căn bản là Transform (Phép biến hình). Để quản lý các GameObject nói chung, sử dụng GameHandler như là GameObject Parent (GameObject cha) dùng để bọc các GameObject con.



Hình 3.1. Giao diện của cây phả hệ GameHandler

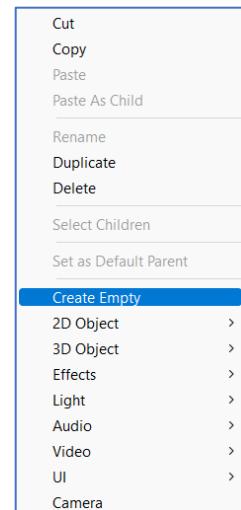
Components của GameHandler không chứa các thuộc tính đặc biệt, chỉ sử dụng như một GameObject cha dùng để quản lý các GameObject con (như Main Camera, UI, Game,...). Xây dựng cây phả hệ và GameHandler là bước đầu trong những dự án Unity để quản lý các GameObject tốt hơn, cài đặt tệp C# script dễ dàng hơn. Tạo GameHandler bằng cách:

- Nhấn chuột phải vào phần Hierarchy và chọn Create Empty
- Đổi tên thành GameHandler
- Chọn các GameObject con và kéo vào GameHandler. Hệ thống sẽ tự động tạo lớp cha với tên là GameHandler.

Cài đặt Camera, Canvas và UI

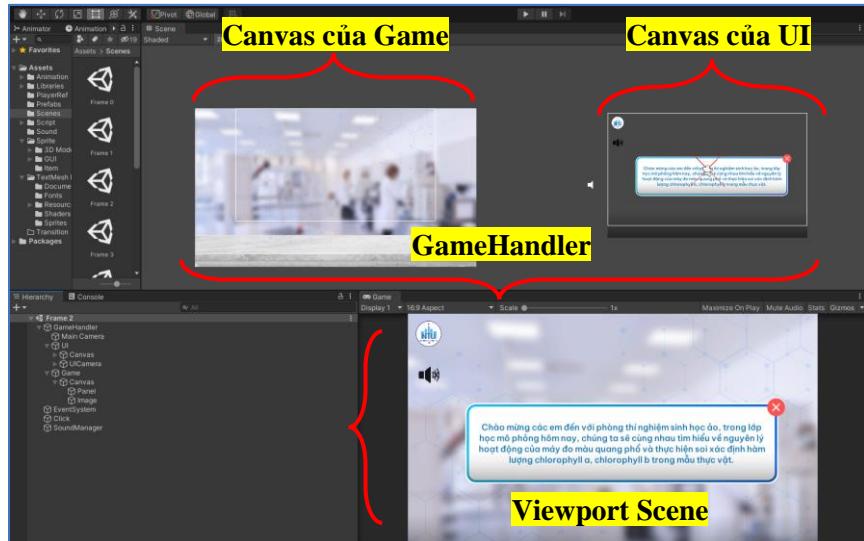
Camera là công cụ tạo định hướng điểm nhìn của người dùng vào chương trình mô phỏng trong môi trường của Unity. Viewport cũng từ đó sinh ra vì Viewport là khu vực mà người dùng có thể nhìn được và giới hạn bởi Camera.

Canvas là một GameObject cha, khu vực chứa các GameObject con liên kết với Camera được giới hạn tầm nhìn. UI tương tự, là một canvas được sử dụng như một lớp



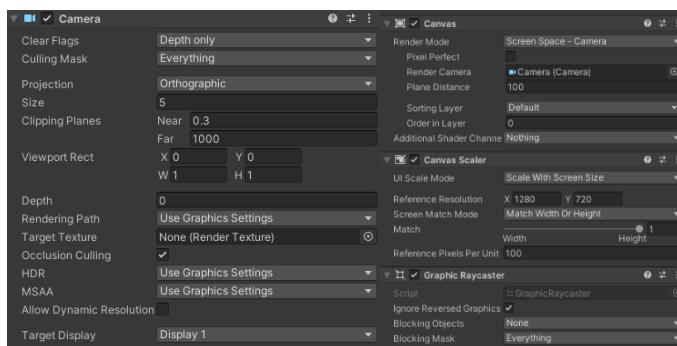
Hình 3.2. Create
Empty để tạo
GameObject mới

phù overlay (chồng lên) toàn bộ GameObject với mục đích là tách biệt tương tác giữa giao diện người dùng và tương tác mô phỏng trong chương trình mô phỏng.



Hình 3.3. Các thành phần đầu tiên của dự án Unity

Để tạo Camera: Nhấp chuột phải vào khu vực cây phả hệ → Chọn Camera. Camera sẽ được chia thành 2 loại tương đương cho 2 Canvas: Canvas của Game và Canvas của UI. Cả hai Camera đều cài đặt mặc định các thuộc tính Canvas (Hình 3.4). Vì đây là chương trình mô phỏng đặt ở góc nhìn 2D nên cần phải điều chỉnh Projection là Orthographic (Phép chiếu trực giao) thay vì Perspective (Phép chiếu xiên) như các dự án 3D. Cả hai Canvas của Game và UI sẽ liên kết phần RenderCamera vào Camera của hai Canvas tương ứng. Trong Canvas Scaler Component của 2 Canvas, điều chỉnh UI Scale Mode thành Scale With Screen Size (Co giãn theo tỉ lệ màn hình). Điều này làm các GameObject trong Viewport sẽ thay đổi kích thước theo tỉ lệ màn hình 16:9.

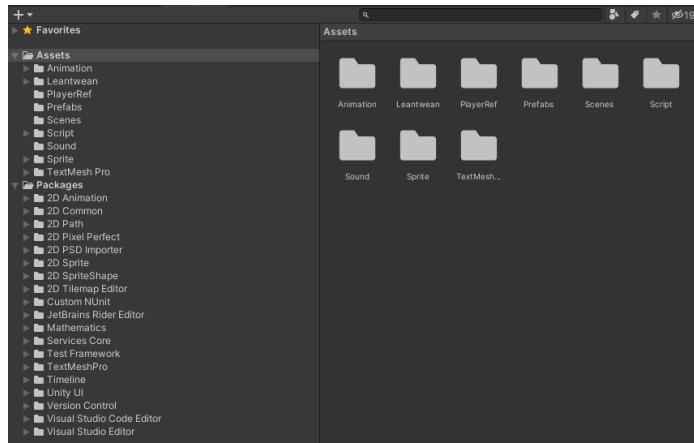


Hình 3.4. Component Camera và Canvas

Hệ thống hóa Assets (Tài nguyên Game)

Tài nguyên Assets sẽ chia thành hai thư mục lớn theo tiêu chuẩn của một dự án Unity. Bao gồm thư mục Assets và Packages. Thư mục Assets sẽ được sử dụng nhiều trong quá trình xây dựng dự án. Thư mục Packages bao gồm toàn bộ thư viện

UnityScript và những thành phần cần thiết để xây dựng nên môi trường mô phỏng của Unity. Thư mục Assets sẽ bao gồm các thư mục con do người dùng khởi tạo. Và trong dự án này sẽ được phân thành các mục như sau (*Hình 3.5*):



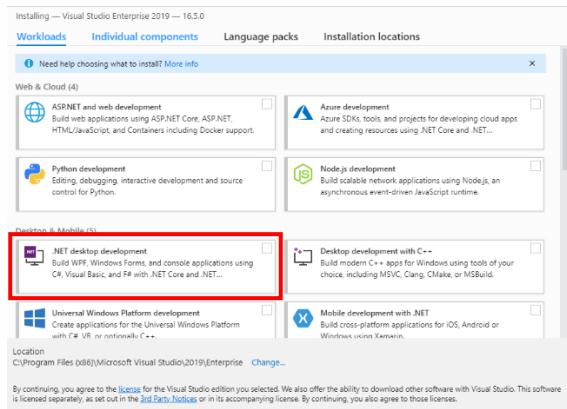
Hình 3.5. Tài nguyên Assets của chương trình mô phỏng

- Animation: Thư mục chứa các dữ liệu về Animation, Animation Controller và Animation Clip.
- LeanTween: Thư viện Animation sử dụng C# để điều chỉnh, dành cho các GameObject cần những animation tùy biến.
- PlayerRef: Thư mục chứa dữ liệu lưu của người dùng.
- Scenes: Thư mục chứa các tệp Scene.
- Script: Thư mục chứa các tệp C# Script của chương trình mô phỏng (animation, savedata, automation,...)
- Sound: Thư mục chứa các tệp tin âm thanh phục vụ cho mục đích giới thiệu, hướng dẫn chương trình bằng giọng nói và hiệu ứng âm thanh của chương trình.
- Sprite: Thư mục lớn bao gồm các file hình ảnh 2D và vật thể 3D để xây dựng cho chương trình mô phỏng.
- TextMesh Pro: Là thư viện tương tác bằng giao diện kéo-thả, được sử dụng để tùy chỉnh phông chữ và tạo hiệu ứng cho chữ.

3.1.2 Cài đặt Visual Studio và C# cho Unity

Cài đặt Visual Studio

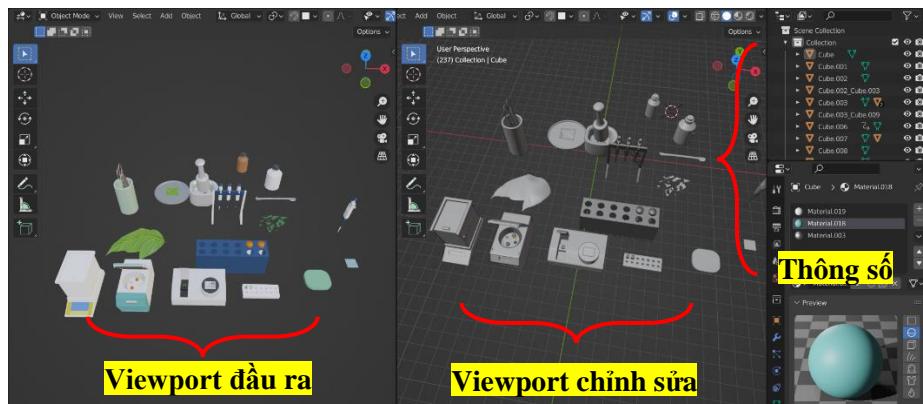
Cài đặt Visual Studio cần lưu ý một điểm nhỏ: Ở phần cài đặt thư viện ngôn ngữ lập trình, chỉ cần tích vào “*.Net desktop development*” và *cài đặt chương trình bằng cách nhấn Install* (*Hình 3.6*). Sau khi đã cài đặt xong Visual Studio, mở tệp C# Script bằng cách: Nhấp chuột phải vào tệp đã chọn → Open C# Project → Visual Studio sẽ tự động mở với tệp C# Script đã chọn (*Hình 3.7*).



Hình 3.6. Giao diện cài đặt Visual Studio
3.1.3 Blender

Cài đặt môi trường Layout

Môi trường mặc định trong Blender chưa phục vụ đúng mục đích của dự án. Do cần xây dựng các vật thể 3D với các điều chỉnh thông số về độ dài, độ cao, độ cong, màu sắc... vì thế, cần điều chỉnh lại các khu vực cửa sổ cho phù hợp và thuận tiện để tối ưu luồng công việc.

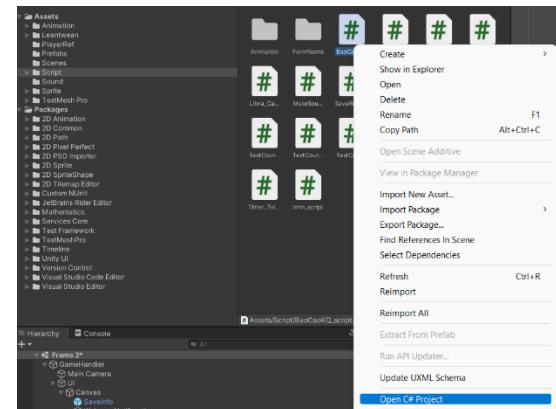


Hình 3.8. Môi trường làm việc của Blender

Không gian làm việc của Blender trong dự án lần này sẽ được chia thành 2 cửa sổ: Viewport đầu ra và Viewport chỉnh sửa (Hình 3.8). Các cửa sổ này sẽ giúp người dùng có góc nhìn trực quan hơn khi “nhào nặn” vật thể 3D. Bên cạnh đó, bên phải màn hình sẽ có phần thông số và material – dùng để khởi tạo các lớp vật liệu cho các mẫu vật 3D.

Cài đặt Export object

Cài đặt Export Object (Kết xuất vật thể) là công đoạn cuối cùng và quan trọng đối với việc kết xuất mô hình 3D để sử dụng trong dự án trò chơi điện tử. Không như các định ảnh 2D thông dụng như PNG, JPEG,... Những vật thể 3D sau khi kết xuất sẽ có những định dạng tiêu biểu như .gltf/glb, .fbx, .obj,... và mỗi định dạng sẽ phục vụ cho các mục đích khác nhau. Tiêu chuẩn GLTF/GLB sẽ được dùng cho các vật thể 3D khi sử dụng trong website hoặc công nghệ AR (Thực tế tăng cường) hoặc các dự án trò chơi điện tử

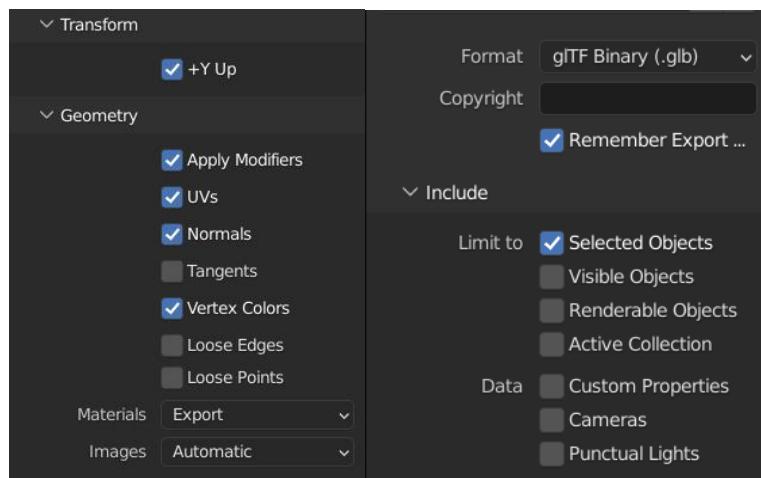


Hình 3.7. Mở C# Script thông qua Unity

Open Scene Additive
View in Package Manager
Import New Asset...
Import Package...
Export Package...
Find References In Scene
Select Dependencies
Refresh
Reimport
Reimport All
Extract From Prefab...
Run API Updater...
Update UXML Schema
Open C# Project

vì cho phép điều chỉnh sâu ở khâu hậu kỳ nhưng bù lại dung lượng tệp sẽ lớn hơn các tiêu chuẩn thông thường. Định dạng .fbx sẽ thường dùng cho các dự án trò chơi điện tử, môi trường VR (Thực tế ảo) vì là định dạng lâu đời, có dung lượng thấp và tính tương thích cao với các Assets của trò chơi điện tử.

Bên cạnh đó, các vật thể 3D sau khi kết xuất đôi khi sẽ không giống hoặc bị lỗi model. Điều đó xảy ra nếu trong quá trình Modeling không điều chỉnh các thông số Scale, Rotation, Position về điểm mặc định. Cùng với đó, ở bước điều chỉnh thông số trong Tùy chọn Export cần phải lựa chọn các thông số đúng (*Hình 3.9*) để có thể xuất ra tệp 3D phù hợp cho quá trình xây dựng chương trình.



Hình 3.9. Các mục điều chỉnh khi Export

Các điểm lưu ý khi xuất các vật thể 3D thành file có đuôi .glb:

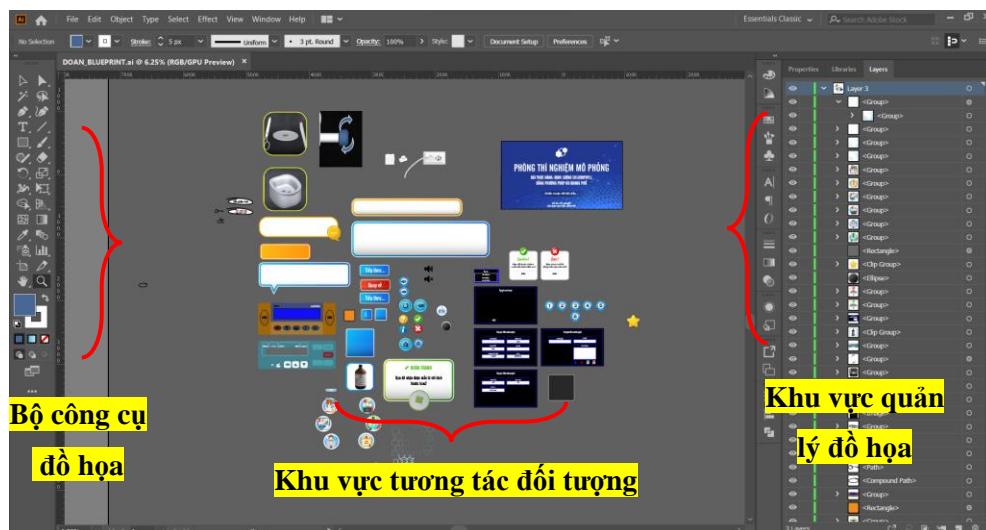
- Ở phần Include: Selected Objects ý nghĩa rằng, chỉ xuất những mô hình 3D được chọn, hạn chế các trường hợp xuất mẫu vật thừa và mẫu vật ẩn.
- Ở phần Geometry: Các tùy chọn về Apply Modifiers (Áp dụng hàm tinh chỉnh), Uvs (Ánh xạ UV), Normals (Hướng vật thể bình thường) và Vertex Colors (Màu nền về dạng RBGs).

3.1.4 Adobe Illustrator

Cài đặt môi trường làm việc

Để tối ưu về nguồn lực, thời gian và công sức, luồng công việc của khâu thiết kế giao diện trong Adobe Illustrator cần được tinh giản tối đa. Và để thực hiện điều đó, không gian làm việc của Adobe Illustrator được phân bổ thành 3 khu vực làm việc chính:

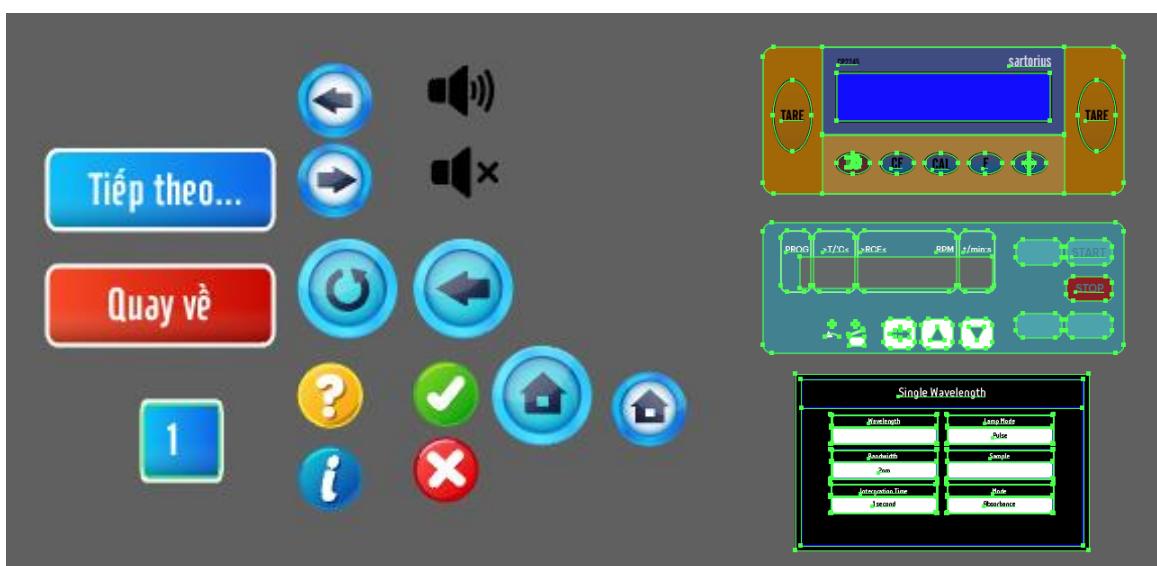
- Bộ công cụ xử lý đồ họa: dùng để tạo Shape 2D, Vector, màu sắc,...
- Khu vực tương tác đối tượng: dùng để tương tác trực tiếp với các đối tượng 2D.
- Khu vực quản lý đồ họa: hiển thị toàn bộ đối tượng đồ họa có trên viewport.



Hình 3.10. Các phần công cụ của Adobe Illustrator

Mỗi khu vực sẽ có vai trò riêng và chỉ quản lý những đối tượng tương đương. Điều đó giúp cho người thiết kế không bị nhầm lẫn giữa các đối tượng và đạt hiệu quả cao trong công việc.

Xây dựng Game Style



Hình 3.11. Icon Button (Bên trái) và Giao diện màn hình các thiết bị (Bên phải)

Game Style (Phong cách đồ họa của trò chơi) có thể xem là hình ảnh đồ họa được dùng để tương tác và hiện thị giao diện chương trình như các Icon, Button,.. (*Hình 3.11*). Để tăng tính thú vị và kích thích cho người dùng, cần sử dụng tông màu sáng và màu sắc tương phản, kèm theo đó sử dụng các tông màu gradient để tạo cảm giác 3D. Bên cạnh đó, chương trình mô phỏng sẽ sử dụng các loại máy như máy ly tâm, cân phân tích, máy đo quang phổ nên cũng phải chuẩn bị phần giao diện màn hình để người dùng có góc nhìn trực quan, rõ ràng về thiết bị (*Hình 3.11*).

3.2 XÂY DỰNG SCENE

3.2.1 Scene 0 - Màn hình chính

Thông tin về Scene

Phân cảnh “Màn hình chính”: khu vực tương đầu tiên của chương trình mô phỏng nên chỉ cần các thông tin hiển thị cần bao hàm đầy đủ chủ đề, tiêu đề của chương trình.

Yêu cầu đặt ra:

- Phần hình ảnh đồ họa chuyên nghiệp, màu sắc phù hợp với chủ đề.
- Nút bấm khởi động tương tác lớn và dễ thấy.

Quy trình xử lý hình ảnh



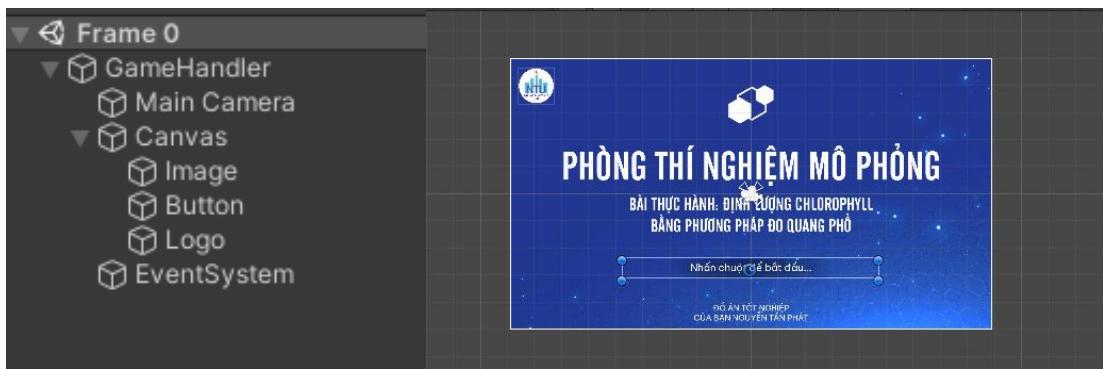
Hình 3.12. Các thành phần của một background (ảnh nền) làm bằng Illustrator

Ở phần ảnh nền sẽ được xây dựng dựa trên thành phần như sau:

- Tên bài thực hành sử dụng Text tool (Công cụ tạo văn bản) và cài đặt phông chữ cho đoạn văn là UTM Cafeta.
- Phần Button (Nút tương tác) có lớp nền màu gradient (Màu lớp 2 màu). Nút tương tác sẽ được tách riêng với phần ảnh nền để sử dụng theo kiểu GameObject để có thể tạo Trigger event (Sự kiện kích hoạt đối tượng) và animation từ trong môi trường Unity.
- Phần nền sau của ảnh nền được tạo ra từ dãy màu gradient có hệ số HEX: (#233c95) và (#3278fd) cùng với họa tiết tinh tế uốn lượn được tạo ra từ công cụ Warp (Uốn cong).

Quy trình xử lý Unity

Ý tưởng: Tạo một GameObject dạng nút tương tác và gán phần nguồn ảnh là hình ảnh nút “Nhấp chuột để bắt đầu...” kèm theo hiển thị với phong cách Animation PopUps (Hoạt họa theo chuyển động Nhấp Lên). Sau đó, sự kiện OnClick() sẽ được kích hoạt và chuyển sang Scene khác khi người dùng nhấp chuột trái vào nút tương tác.



Hình 3.13. Cây phả hệ (Bên trái) và Giao diện đồ họa phân cảnh 0 (Bên phải)

Ở phân cảnh “*Màn hình chính*”, cần lưu ý đến 2 Trigger event của nút tương tác.

Sự kiện của Button sẽ được xây dựng bao gồm 2 thành phần như sau:

- Animation của nút tương tác khi vừa khởi động chương trình
- Trigger event khi người dùng nhấp chuột trái vào khu vực được lập trình để chuyển sang phân cảnh tiếp theo

Tạo C# Script để cài đặt Animation tùy chỉnh cho đối tượng Button

```
using UnityEngine;
using Utils.Tweens.Data;

namespace Utils.Tweens
{
    public class LeanTweenerAnimation : MonoBehaviour
    {
        [Header("Open Animation Config")]
        [SerializeField] private TweenAnimationData openAnimationData = new
        TweenAnimationData(LeanTweenType.easeOutBack);

        [Header("Close Animation Config")]
        [SerializeField] private TweenAnimationData closeAnimationData = new TweenAnimationData(LeanTweenType.easeInBack);
```

Ở hai phần Header(), được sử dụng để tạo phần khai báo đối tượng ở dạng Input.

Được sử dụng để thao tác trong phần lập trình giao diện kéo-thả của Unity. Và tiếp theo đó là khai báo đối tượng Animation dựa trên thư viện Tweens có sẵn.

```
private Vector3 initialSize;
private void Awake()
{
    GetInitialSize();
}
private void OnEnable()
{
    OpenAnimation();
}
public void OpenAnimation()
{
    if (openAnimationData.EaseType == LeanTweenType.notUsed) return;

    transform.localScale = Vector3.zero;
    LeanTween.scale(GameObject, initialSize, openAnimationData.Duration)
        .setDelay(openAnimationData.Delay).setEase(openAnimationData.EaseType)
        .setOnComplete(() → openAnimationData.OnAnimationComplete?.Invoke());
}
```

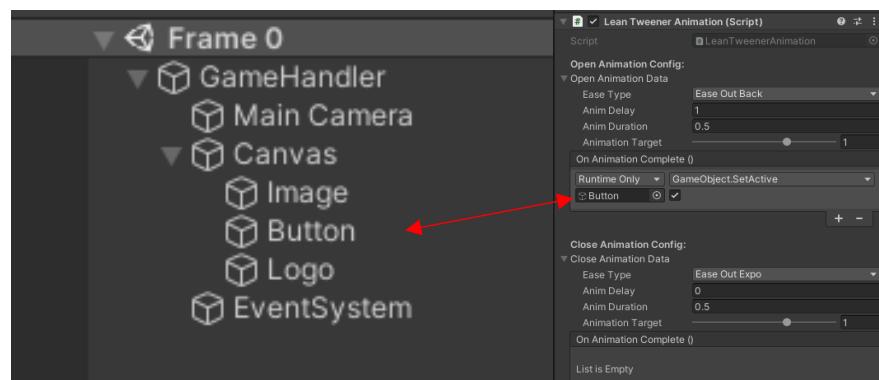
OpenAnimation(): được sử dụng để kích hoạt animation khi đối tượng gán vào ở trạng thái “Mở” animation theo dạng PopUps - Các thông số được tùy biến sẽ bao gồm

phép Scale (Điều chỉnh kích thước), chỉnh độ Delay (độ trễ) của Animation và có thể được điều chỉnh bằng số trong giao diện lập trình của Unity.

```
public void CloseAnimation()
{
    if (closeAnimationData.EaseType == LeanTweenType.notUsed) return;
    LeanTween.scale(GameObject, Vector3.zero, closeAnimationData.Duration)
        .setDelay(closeAnimationData.Delay).setEase(closeAnimationData.EaseType)
        .setOnComplete(ToDoAfterCloseAnimation);
}
private void ToDoAfterCloseAnimation()
{
    ResetCardSize();
    closeAnimationData.OnAnimationComplete?.Invoke();
}
private void GetInitialSize() => initialSize = transform.localScale;
private void ResetCardSize() => transform.localScale = initialSize;
}
```

CloseAnimation(): được sử dụng để kích hoạt animation khi đối tượng gán vào ở trạng thái “Đóng” animation theo dạng PopUps - Các thông số được tùy biến sẽ bao gồm phép Scale (Điều chỉnh kích thước), chỉnh độ Delay (độ trễ) của Animation và có thể được điều chỉnh bằng số trong giao diện lập trình của Unity.

ToDoAfterAnimation(): hàm thực thi ResetCardSize() với điều kiện CloseAnimation() đã được thực thi. Sau khi kiểm tra, điều kiện = true, ResetCardSize() sẽ được kích hoạt với sự điều chỉnh lại phép Scale của đối tượng trở về mặc định.



Hình 3.14. Gán GameObject Button vào C# Script LeanTween

Sau khi lập trình xong, phần C# Script được gán vào Components của Button (Hình 3.14). Ở phần khai báo trong Components, chọn đối tượng Button (Nút tương tác cần được cài đặt Trigger event). Và sau khi gán thành công, nút tương tác có thể được điều chỉnh độ Delay (độ trễ), Duration (Độ chờ),... Cuối cùng, cài đặt giá trị cho Button là “GameObject.SetActive”, điều này có nghĩa là bật trạng thái “Mở” cho nút tương tác, từ đó, kích hoạt Animation khi khởi động phân cảnh.

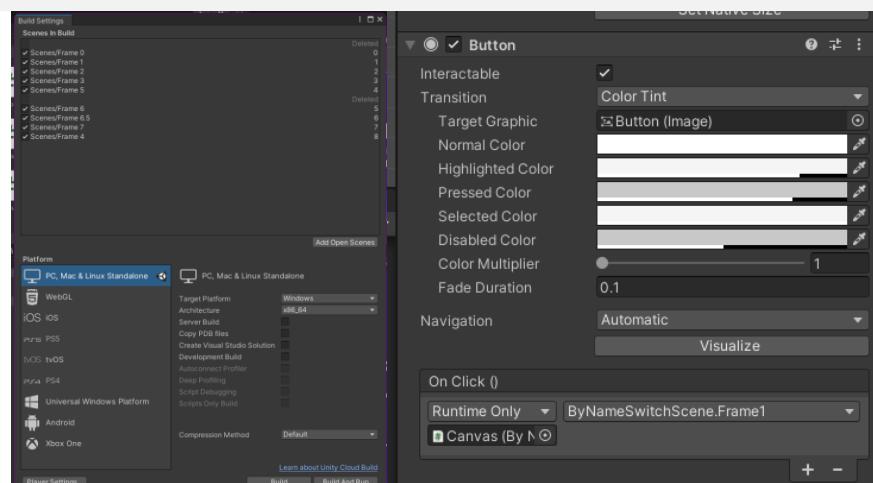
Tiếp theo là sự kiện Trigger Scene cho Button:

```
public class ByNameSwitchScene : MonoBehaviour
{
    public void Frame1()
    {
```

```

        SceneManager.LoadScene("Frame 1");
    }
    public void Frame2()
    {
        SceneManager.LoadScene("Frame 2");
    }
    public void Frame3()
    {
        SceneManager.LoadScene("Frame 3");
    }
    public void Frame4()
    {
        SceneManager.LoadScene("Frame 4");
    }
    public void Frame5()
    {
        SceneManager.LoadScene("Frame 5");
    }
    public void Frame5_5()
    {
        SceneManager.LoadScene("Frame 5.5");
    }
    public void Frame6()
    {
        SceneManager.LoadScene("Frame 6");
    }
    public void Frame6_5()
    {
        SceneManager.LoadScene("Frame 6.5");
    }
    public void Frame7()
    {
        SceneManager.LoadScene("Frame 7");
    }
}

```



Hình 3.15. Cập nhật Scene bằng thư viện SceneManagement (Bên trái) và chọn hàm OnClick() để kích hoạt Scene (Bên phải)

Để chuyển sang phân cảnh khác, sử dụng thư viện SceneManagement của Unity. Sau đó, gọi hàm `SceneManager.LoadScene("...")` có ý nghĩa là chuyển từ Scene hiện tại qua Scene theo yêu cầu (`SceneManager.LoadScene("Frame 5")`). Và ở giao diện làm việc của Unity, để thêm các phân cảnh vào thư viện Quản lý Scene của Unity: Chọn File → Tích chọn các Scene đang sử dụng → Build Setting (*Hình 3.15*).

Tương tự như Animation, C# Script “`ByNameSwitchScene`” được gán vào nút tương tác cần được kích hoạt `OnClick()` (*Hình 3.15*). Ở phần tùy chọn, `ByNameSwitchScene.Frame1`: chuyển sang Scene 1 khi Click vào nút tương tác.

3.2.2 Scene 1 – Đăng nhập

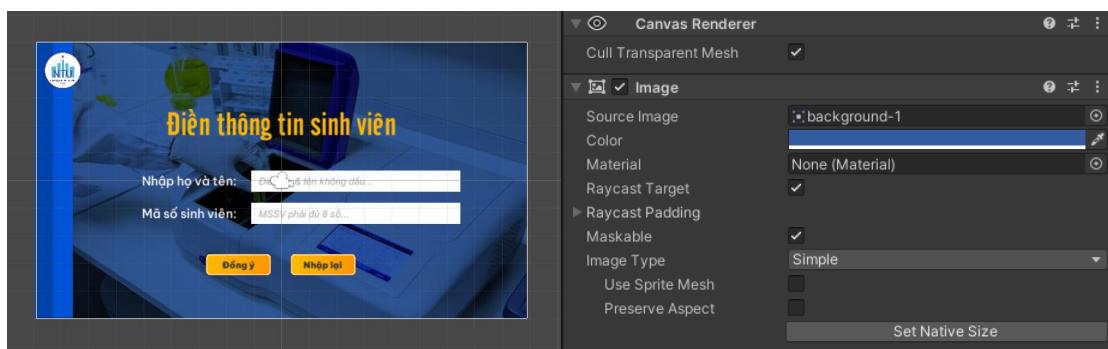
Thông tin về Scene

Phân cảnh 1 là khu vực để người dùng điền thông tin về tên sinh viên và mã số sinh viên. Để truy cập vào chương trình mô phỏng, vùng điền thông tin của “Họ và tên” và “Mã số sinh viên” cần được kiểm tra hợp lệ.

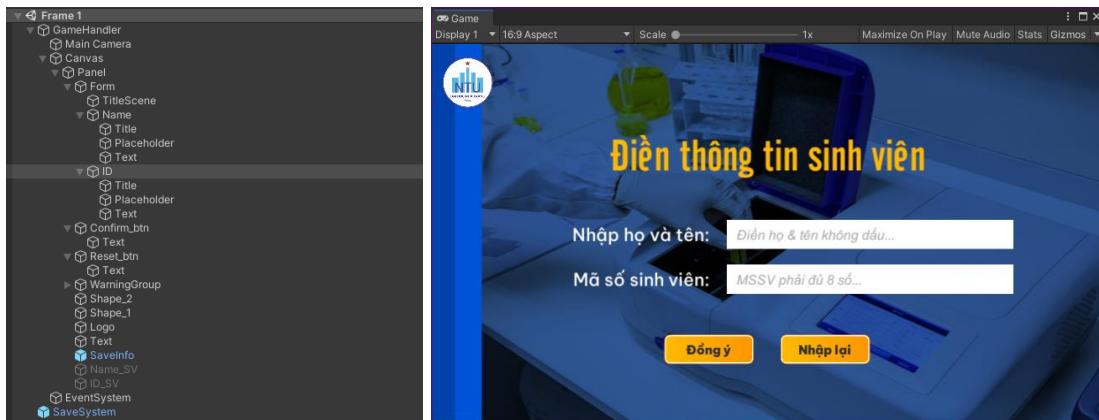
Yêu cầu đặt ra:

- Người dùng đăng nhập theo dạng Form
- Có tính năng kiểm tra họ tên và mã số sinh viên hợp lệ
- Lưu lại thông tin dùng sau khi đăng nhập thành công

Quy trình xử lý hình ảnh



Hình 3.16. Các thành phần của Phân cảnh 1 (Bên trái) và Canvas Renderer (Bên phải)



Hình 3.17. Cây phả hệ (Bên trái) và Giao diện phân cảnh “Đăng nhập” (Bên phải)

Ảnh nền và các thành phần ảnh được xử lý bằng giao diện đồ họa của Unity. Trong đó, ảnh nền sẽ được gán vào GameObject Panel có thuộc tính Canvas Render. Từ đó, gán nguồn ảnh vào Source Image. Tùy chỉnh phần Color là màu xanh dương đậm và giảm độ Opacity để tạo ra hiệu ứng ám mờ như trong Hình 3.16. Các vùng nhập dữ liệu sẽ được khởi tạo bằng đối tượng GameObject InputField.

Quy trình xử lý Unity

Ý tưởng: Hai vùng nhập thông tin về Họ và tên, Mã số sinh viên sẽ thay bằng đối

tương GameObject InputField có Components InputField dùng để nhập liệu và cài đặt các thông số để giới hạn khả năng nhập liệu. Có hai nút tương tác là Button “Đồng ý” và Button “Nhập lại”. Cả hai nút sẽ được cài đặt Trigger event để thực thi các yêu cầu kiểm tra dữ liệu. Phân cảnh sẽ có hai Trigger event:

- Xác nhận thông tin Đăng Nhập
- Cài đặt lại thông tin nhập liệu

Xác nhận thông tin Đăng nhập và cài đặt mặc định thông tin:

```
public class GetInputField : MonoBehaviour
{
    public GameObject WarningText_2;
    public GameObject WarningText;
    public InputField usernameInput;
    public InputField idInput;

    public void CheckValid()
    {
        if (string.IsNullOrEmpty(usernameInput.text) || string.IsNullOrEmpty(idInput.text))
        {
            WarningText.GameObject.SetActive(true);
            WarningText_2.GameObject.SetActive(false);
        }
        else if (idInput.text.Length < 8)
        {
            WarningText_2.GameObject.SetActive(true);
            WarningText.GameObject.SetActive(false);
        }
        else
        {
            SceneManager.LoadScene("Frame 2");
        }
    }
}
```

CheckValid() dùng If else lồng nhau để kiểm tra tính đúng đắn của 2 InputField. Nếu một trong hai hoặc cả hai InputField bị trống, thông báo “Chưa điền đủ thông tin” được thực thi (*Hình 3.18*). Bên cạnh đó, hàm sẽ kiểm tra xem InputField của “Mã số sinh viên” đã đủ 8 chữ số. Nếu chưa đạt theo yêu cầu, thông báo “Mã số sinh viên chưa hợp lệ” sẽ hiện lên. Khi đã thỏa mãn tất cả điều kiện If-else, chương trình sẽ tự động chuyển sang phân cảnh 2 và lưu lại thông tin người dùng.



Hình 3.18. Thông báo khi InputField sai thông tin (Bên trái) và bỏ trống (Bên phải)

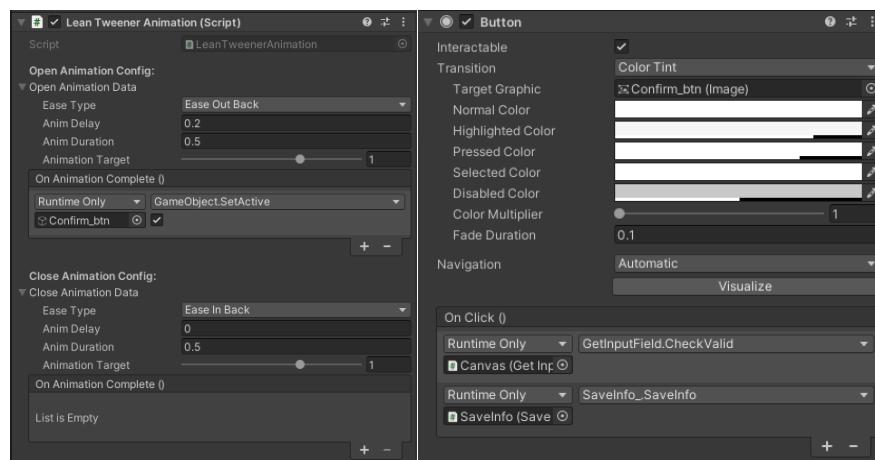
```
public void ResetField()
{
    if (!string.IsNullOrEmpty(usernameInput.text) || !string.IsNullOrEmpty(idInput.text))
    {
        usernameInput.text = "";
        idInput.text = "";
    }
}
```

ResetField() được dùng để xóa thông tin đã trên trong InputField. Được thực thi khi người dùng thực thi OnClick() của Button “Nhập lại”.

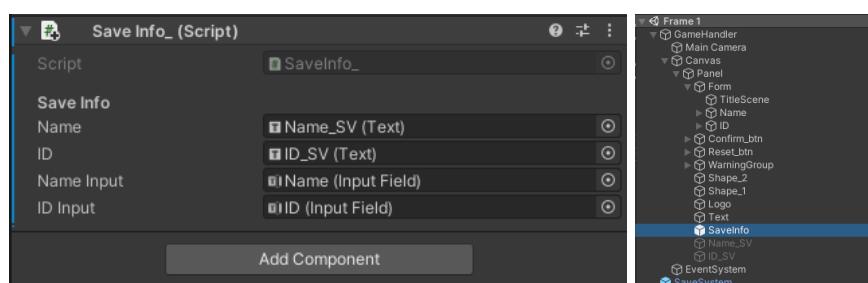
Lưu lại thông tin đăng nhập:

```
public class SaveInfo_ : MonoBehaviour
{
    [Header("Save Info")]
    public Text Name;
    public Text ID;
    public InputField NameInput;
    public InputField IDInput;
    public void LoadInfo()
    {
        Name.text = "Name: " + SaveSystem.GetString("TextName");
        ID.text = "ID: " + SaveSystem.GetString("TextID");
    }
    public void SaveInfo()
    {
        Name.text = NameInput.text;
        ID.text = IDInput.text;
        SaveSystem.SetString("TextName", Name.text);
        SaveSystem.SetString("TextID", ID.text);
    }
}
```

SaveInfo() sử dụng biến đã nhập từ InputField và gán vào một đối tượng mới. Thư viện SaveSystem của Unity để lưu các thông tin dưới dạng String. LoadInfo() sẽ gán biến vào đối tượng cần truyền thông tin vào Text (GameObject) ở phân cảnh cần dùng.



Hình 3.19. Cài đặt hàm Animation cho Button (Bên trái) và CheckValid() (Bên phải)



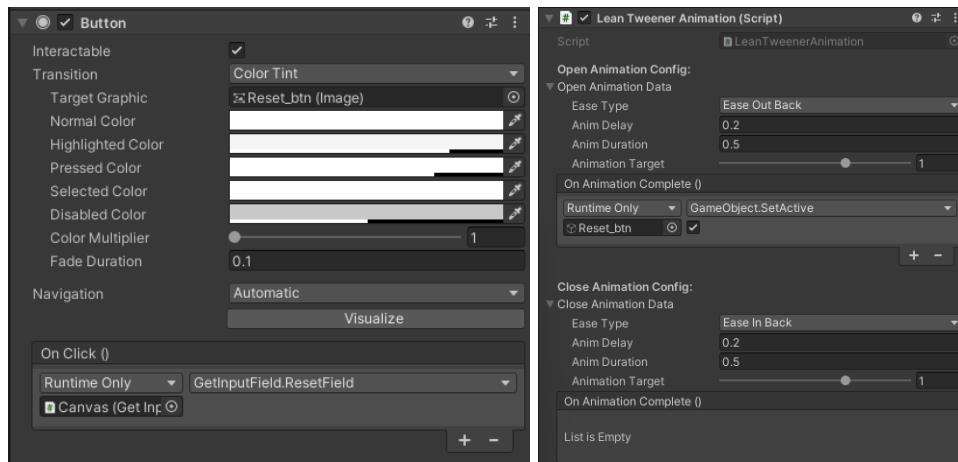
Hình 3.20. Khai báo các đối tượng cần lưu lại thông tin (Bên trái) và GameObject tượng trưng cho C# Script (Bên phải)

Ở phần Components của Button “Đóng ý”, cài đặt animation bằng thư viện Lean Tweener và chỉnh GameObject.SetActive cho đối tượng để bật trạng thái “Mở” cho đối tượng. Cài đặt OnClick() cho nút tương tác “Đóng ý” hai hàm: SaveInfo() để lưu dữ liệu

và hàm CheckValid() để kiểm tra các yêu cầu vùng nhập liệu.

C# Script “SaveInfo” sẽ được gán vào một GameObject được sử dụng để tương ứng cho C# Script dưới dạng GameObject và dùng để khai báo các đối tượng InputField được lưu riêng biệt vào hệ thống

Tương tự, phần Components (Button “Reset”), gán ResetField() thực thi hàm khi OnClick() và thư viện LeanTweener để thực thi Animation khi khởi động (*Hình 3.21*).

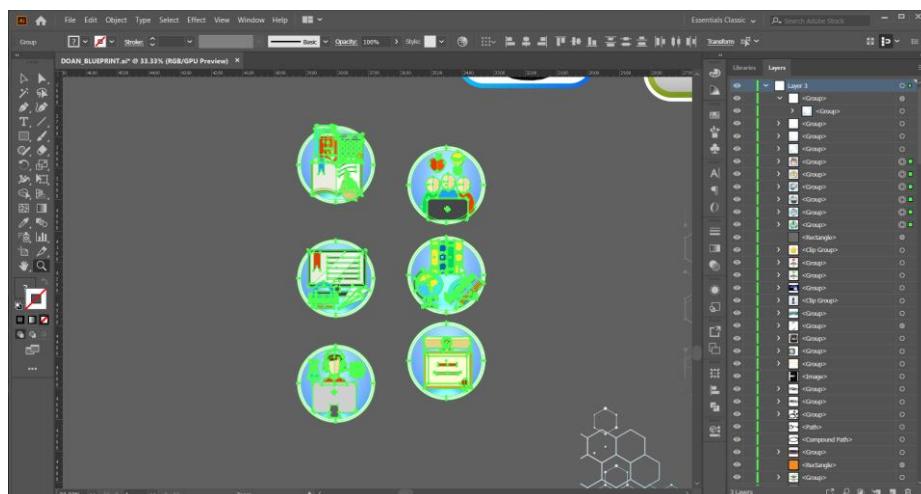


Hình 3.21. Cài đặt hàm ResetField() cho Button (Bên trái) và LeanTweener (Bên phải)

3.2.3 Scene 2 – Menu

Thông tin về Scene

Phân cảnh 2 là khu vực người dùng xem tổng quan các khu vực tương tác khác của chương trình mô phỏng. Một số khu vực chưa thể tương tác được và để kích hoạt, người dùng cần phải hoàn thành những khu vực có thể tương tác được.



Hình 3.22. Các thành phần của phân cảnh “Menu”

Yêu cầu đặt ra:

- Có thông báo chào mừng người dùng và sử dụng giọng nói AI.
- Có tính năng tắt/bật âm thanh

- Hiển thị danh sách phân cảnh
- Hiển thị thông tin người dùng thực hiện (Cửa sổ đăng nhập)

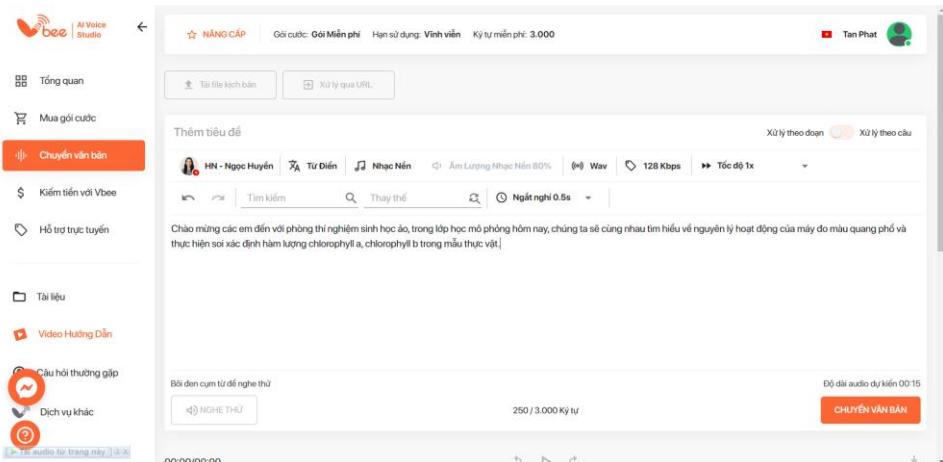
Quy trình xử lý hình ảnh

Các biểu tượng của danh sách Menu trong phân cảnh 2 được thiết kế bằng Adobe Illustrator với các công cụ như Pen tool, Shape tool, Gradient color.

Quy trình xử lý âm thanh giọng nói AI

Chuẩn bị đoạn văn giới thiệu từ trình soạn thảo văn bản:

“Chào mừng các em đến với phòng thí nghiệm sinh học ảo, trong lớp học mô phỏng hôm nay, chúng ta sẽ cùng nhau tìm hiểu về nguyên lý hoạt động của máy đo màu quang phổ và thực hiện soi xác định hàm lượng chlorophyll a, chlorophyll b trong mẫu thực vật.”



Hình 3.23 Gán đoạn văn vào phần nhập liệu của công cụ chuyển đổi âm thanh Vbee.vn

TIÊU ĐỀ	KÝ TỰ	THỜI GIAN	TRẠNG THÁI	GỌNG ĐỌC	HÀNH ĐỘNG
Chào mừng các e...	250	29-06-2022 18:12	<div style="width: 10%;">10%</div>	HN - Ngọc Huyền	
Chất diệp lục chlor...	414	29-03-2022 14:13	Thành công	SG - Thảo Trinh	
Chất diệp lục chlor...	414	29-03-2022 14:12	Thành công	HN - Ngọc Huyền	
Chào mừng các e...	197	28-03-2022 23:23	Thành công	SG - Thảo Trinh	
Chào mừng các e...	407	28-03-2022 22:48	Thành công	SG - Thảo Trinh	

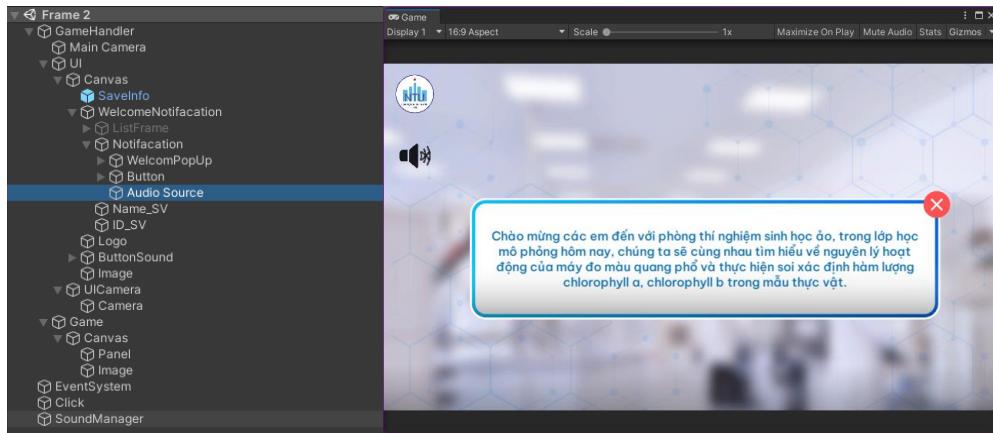
Hình 3.24 Bấm “Tải về” khi trạng thái đạt 100%

Truy cập vào Vbee.vn và lựa chọn công cụ chuyển đổi âm thanh Studio. Gán đoạn văn vào khu vực nhập liệu, lựa chọn giọng nói và bấm “Chuyển văn bản” (Hình 3.23).

Đợi chương trình xử lý, khi đạt trạng thái 100%, bấm vào biểu tượng “Tải về” để tải tệp âm thanh về với định dạng .mp3 (Hình 3.24).

Quy trình xử lý Unity

Ý tưởng: Khi Scene 2 khởi động, thông báo chào mừng được hiển thị theo dạng Animation PopUps của thư viện LeanTweener, kèm theo giọng đọc AI. Nút tương tác có chức năng tắt/bật âm thanh được bố trí bên cạnh logo của Trường Đại học Nha Trang. Danh sách menu sẽ được hiển thị theo thứ tự.



Hình 3.25. Cây phả hệ của phân cảnh 2 (Bên trái) và Thông báo chào mừng (Bên phải)

Khởi tạo GameObject “WelcomeNotification” và hai GameObject con bao gồm:

- GameObject “ListFrame”
- GameObject “Notifacation”

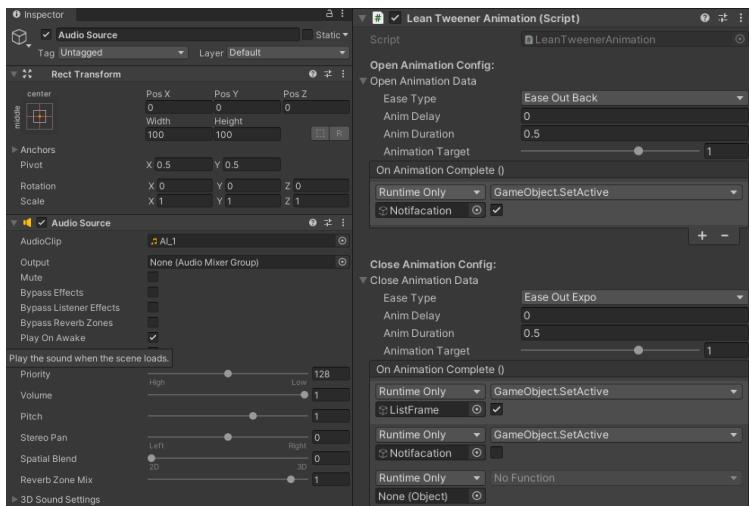
Cài đặt âm thanh và Animation cho thông báo chào mừng

GameObject “WelcomPopUp” trong Notifacation (*Hình 3.25*) sẽ được gán thư viện LeanTweener để hiện thị thông báo theo dạng PopUps khi Scene 2 khởi động. Bên cạnh đó, khởi tạo GameObject mang thuộc tính AudioSource và gán tệp âm thanh vào Components Audio Clip (*Hình 3.26*). Ở cài đặt AudioSource, chọn tích “Play On Awake”, tính năng được dùng phát ra âm thanh ngay khi khởi động Scene.

Thư viện Unity có hỗ trợ tắt/bật âm thanh bằng hàm “

```
public class ExampleClass : MonoBehaviour
{
    AudioSource audioSource;
    void Start()
    {
        audioSource = GetComponent<AudioSource>();
    }

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space))
            audioSource.mute = !audioSource.mute;
    }
}
```



Hình 3.26. Gán âm thanh vào AudioClip (Bên trái) và nút “Tắt thông báo” (Bên phải)

Tuy nhiên, hàm này chỉ áp dụng cho một AudioSource và một phân cảnh nhất định. Vì vậy hàm này không thể sử dụng trong trường hợp: Một phân cảnh có nhiều đối tượng AudioSource hoặc khi người dùng chuyển sang phân cảnh khác. Để giải quyết vấn đề này, cần khởi tạo C# Script để lập trình một số dữ liệu liên quan tới quản lý thông tin của người dùng và quản lý đa phân cảnh:

```
public class SoundManager : MonoBehaviour
{
    [SerializeField] Image soundOnIcon;
    [SerializeField] Image soundOffIcon;
    private bool muted = false;
    void Start()
    {
        if (!PlayerPrefs.HasKey("muted"))
        {
            PlayerPrefs.SetInt("muted", 0);
            Load();
        }
        UpdateButtonIcon();
        AudioListener.pause = muted;
    }
```

Start() tự động thực thi khi khởi động Scene và kiểm tra dữ liệu của PlayerPrefs đã được gán biến “*Muted*” bằng việc người dùng nhấn vào nút tương tác “*Bật/tắt âm thanh*”. Nếu chưa gán biến sẽ tự động cập nhật biểu tượng cho nút tương tác “*Bật/tắt âm thanh*” bằng hàng UpdateButtonIcon().

```
public void OnButtonPress()
{
    if (muted == false)
    {
        muted = true;
        AudioListener.volume = 0;
    }
    else
    {
        muted = false;
        AudioListener.volume = 1;
    }
    Save();
    UpdateButtonIcon();
}
```

Ở hàm OnButtonPress(), trong đó gán giá trị true/false cho biến “*Muted*” tùy vào

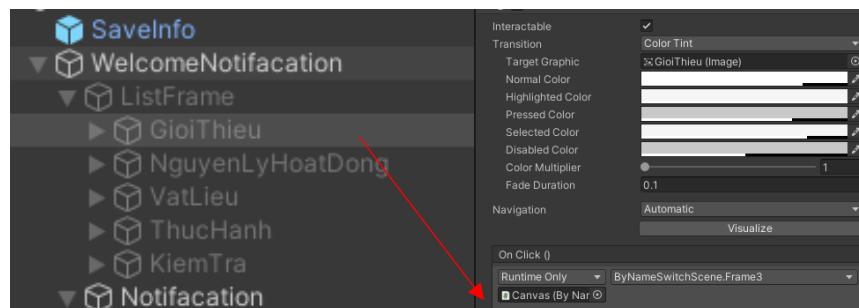
giá trị biến ban đầu của “*Muted*”. Trong Components AudioListener, có thuộc tính tên “*Volume*”. *Volume* = 1, có nghĩa là nguồn âm thanh sẽ bật, *Volume* = 0 thì nguồn âm thanh sẽ tắt. Vì vậy, sau khi nhận trạng thái ở thuộc tính *Volume*, hàm *Save()* được thực thi và lưu lại giá trị cho biến “*Muted*”.

```
private void UpdateButtonIcon()
{
    if (muted == false)
    {
        soundOnIcon.enabled = true;
        soundOffIcon.enabled = false;
    }
    else
    {
        soundOnIcon.enabled = false;
        soundOffIcon.enabled = true;
    }
}
```

UpdateButtonIcon() được sử dụng để đổi biểu tượng âm lượng tương ứng với trạng thái tắt/bật của âm thanh.

```
private void Load()
{
    muted = PlayerPrefs.GetInt("muted") == 1;
}
private void Save()
{
    PlayerPrefs.SetInt("muted", muted ? 1 : 0);
}
```

Trigger event cho nút “Chuyển cảnh”



Hình 3.27. Gán hàm chuyển Scene cho GameObject

Trong GameObject “*ListFrame*”, sẽ có 5 nút tương tác tương ứng cho 5 phân cảnh có thể tương tác. Mỗi GameObject sẽ có C# Script chuyển sang Scene khác tương ứng (Hình 3.27). Thuộc tính “*SetActive(true)*” khi người dùng hoàn thành Scene trước.

3.2.4 Scene 3 – Giới thiệu

Thông tin về Scene

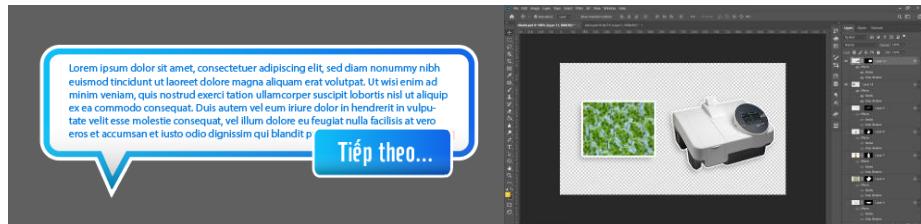
Phân cảnh 3: giới thiệu người dùng về những khái niệm tổng quan của chương trình mô phỏng “*Định lượng Chlorophyll bằng phương pháp đo quang phổ*”.

Yêu cầu đặt ra:

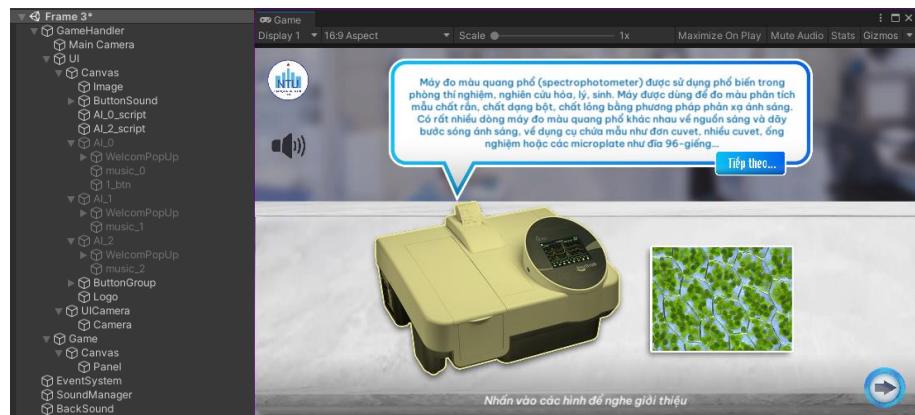
- Hiển thị phân thông tin ở dạng chữ và có giọng đọc AI kèm theo.
- Hình ảnh đồ họa hiện thị rõ chủ đề của chương trình.

Quy trình xử lý hình ảnh

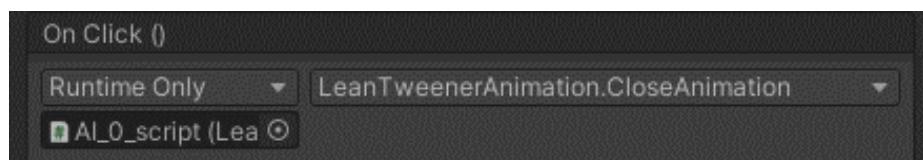
Hình ảnh trong phân cảnh 3 bao gồm: Text GameObject và Button GameObject. Phần văn bản trong Text GameObject được gán thành đối tượng hình ảnh để tránh tình trạng lỗi phông chữ do Unity chưa hỗ trợ Tiếng Việt. Bên cạnh đó, các hình ảnh về những đối tượng đồ họa chủ được tạo hiệu ứng đổ bóng và viền trắng bằng Adobe Photoshop (Hình 3.28) để nổi bật vật thể trong phân cảnh.



Hình 3.28. Template thông báo của Scene “Giới Thiệu” (Bên trái) và Tạo hiệu ứng cho đối tượng đồ họa bằng Adobe Photoshop (Bên phải)



Hình 3.29. Cây phả hệ (Bên trái) và Khung hình đầu của Phân cảnh 3 (Bên phải)



Hình 3.30. Gán thư viện Animation cho GameObject

Quy trình xử lý Unity

Ý tưởng: Phân cảnh 3 sẽ bao gồm những khái niệm chính được soạn sẵn và kèm theo giọng đọc AI. Các GameObject này sẽ được thực thi khi OnClick() của hai Button hình ảnh được kích hoạt. Các hình ảnh sẽ được khởi tạo dưới dạng Button GameObject, trở thành các hình ảnh có thể tương tác được như Button GameObject thông thường.

Ở mỗi Button hình ảnh, cài đặt thư viện LeanTweener để thực thi animation dạng PopUps khi các nút tương tác có Trigger event (Hình 3.30). Khi các Button được nhấn vào bởi người dùng, bảng thông tin sẽ được hiển thị như Hình 3.29.

3.2.5 Scene 4 – Nguyên lý hoạt động của máy đo quang phổ

Thông tin về Scene

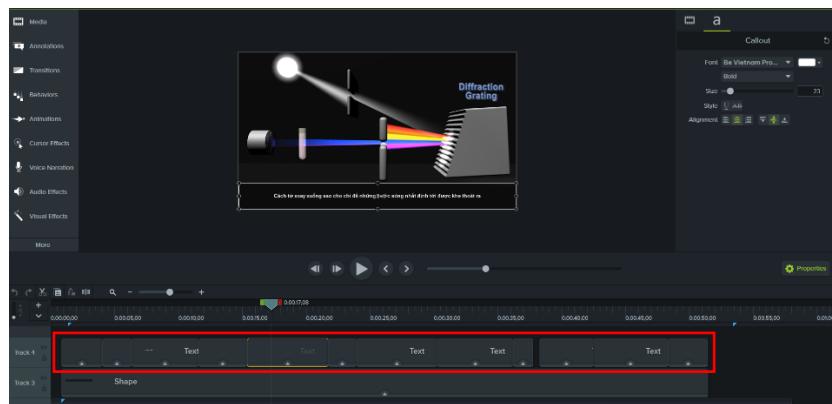
Phân cảnh 4 là khu vực người dùng được tìm hiểu nguyên lý hoạt động cơ bản của máy đo quang phổ và khái niệm mật độ quang học.

Yêu cầu đặt ra:

- Video sẽ được chạy tự động khi vừa chuyển Scene
- Video không sử dụng thanh tua/chạy/dừng mà chỉ phát từ đầu đến cuối video
- Có phụ đề Tiếng Việt
- Có nút để tắt/bật âm thanh.
- Chuyển Scene khác với điều kiện đã phát hết video

Quy trình xử lí video

Video sử dụng trong chương trình mô phỏng được tải về từ NCBioNetwork (<https://www.ncbiornetwork.org/educational-resources/videos/how-does-spectrophotometer-work>). Video gốc chưa hỗ trợ Tiếng Việt và tốc độ chạy chậm. Vì vậy, để xử lý các vấn đề trên, sử dụng phần mềm chỉnh sửa video chuyên dụng tên là Camtasia. Video sẽ được tăng tốc độ phát 1.5x và chèn việt ngữ (Hình 3.31).



Hình 3.31. Giao diện xử lí Camtasia



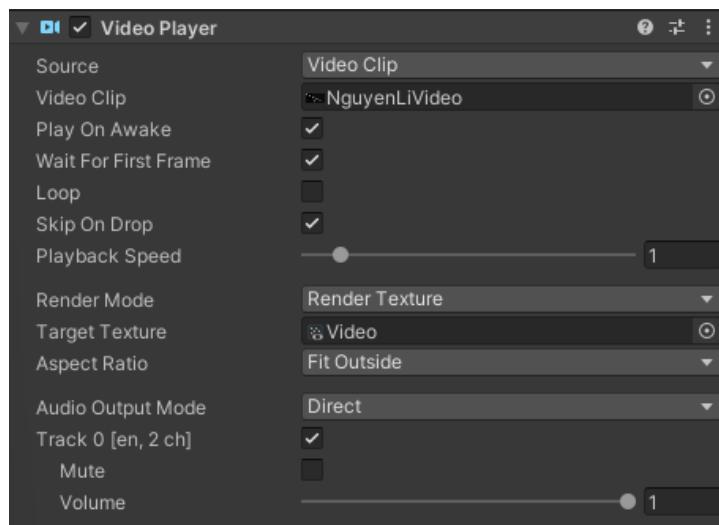
Hình 3.32. Cây phả hệ (Bên trái) và Giao diện đồ họa của phân cảnh 4 (Bên phải)

Quy trình xử lí Unity

Ý tưởng: Sử dụng các thuộc tính có sẵn của Video Player để điều chỉnh video theo yêu cầu. Khi video kết thúc, nút tương tác “Chuyển Scene” sẽ được kích hoạt bằng cách lập trình C# Script để tạo độ Delay (độ trễ) của thuộc tính “Interactable” của Button.

Như các phân cảnh trước, phân cảnh 5 sẽ có giao diện UI chứa nút “Bật/tắt âm thanh”. Bên cạnh đó, khởi tạo GameObject cha có tên “Video_Group” bao gồm các thành phần của video. Trong Components Video Player, cần cài đặt một số tùy chỉnh sau (*Hình 3.33*):

- Play On Awake: dùng để tự chạy video khi Scene được khởi động
- Wait For First Frame: video sẽ phát sau khi các GameObject được thực hiện khung hình đầu tiên (có thể đổi tương đương với 0,1 giây)
- Play Speed = 1: tốc độ video chạy ở mức độ bình thường
- Volume = 1: âm lượng của video được phát ở mức độ mặc định



Hình 3.33. Điều chỉnh thuộc tính Video Player cho GameObject

Tạo sự kiện kích hoạt nút tương tác “Chuyển Scene” khi video kết thúc

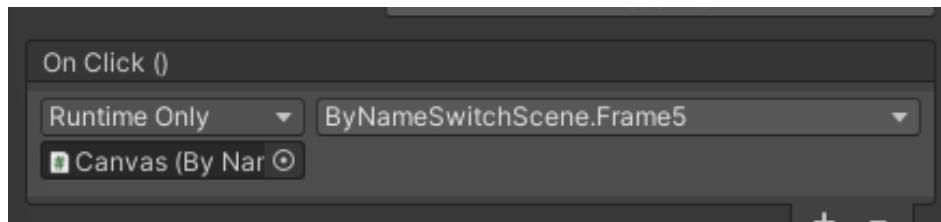
```
public class Button_DelayInteracter : MonoBehaviour
{
    private Button Btn_;

    void Start()
    {
        Btn_ = GameObject.GetComponent<Button>();
        Btn_.interactable = false;
        Invoke("Open", 59);
    }
    private void Open()
    {
        Btn_.interactable = true;
    }
}
```

Start() của thư viện Unity có thuộc tính mặc định là tự động thực thi khi khởi động Scene. Trong hàm Start(), khởi tạo biến được dùng để gán đối tượng “`GameObject.GetComponent<Button>()`”. Đối tượng Button được gán vào sẽ thực thi

thuộc tính Interactable trong 59 giây. Hàm Open() được thực thi và gán giá trị True cho thuộc tính Interactable của đối tượng Button.

Invoke() là hàm của thư viện của Unity được sử dụng để gia hạn thời gian thực thi đối tượng. Có hai thành phần là: 1. Tên hàm; 2. Thời gian độ trễ thực hiện. Sau khi khởi tạo hàm Open(), gán giá trị True cho thuộc tính Interactable của biến Btn_. Và gắn vào Invoke() để chúng thực hiện sau 59 giây, tương đương thời lượng của video.



Hình 3.34. Gán hàm chuyển Scene cho Button

Bên cạnh đó, cài đặt sự kiện OnClick() “Chuyển Scene” cho nút tương tác bằng C# Script đã viết trước đó (Hình 3.34).

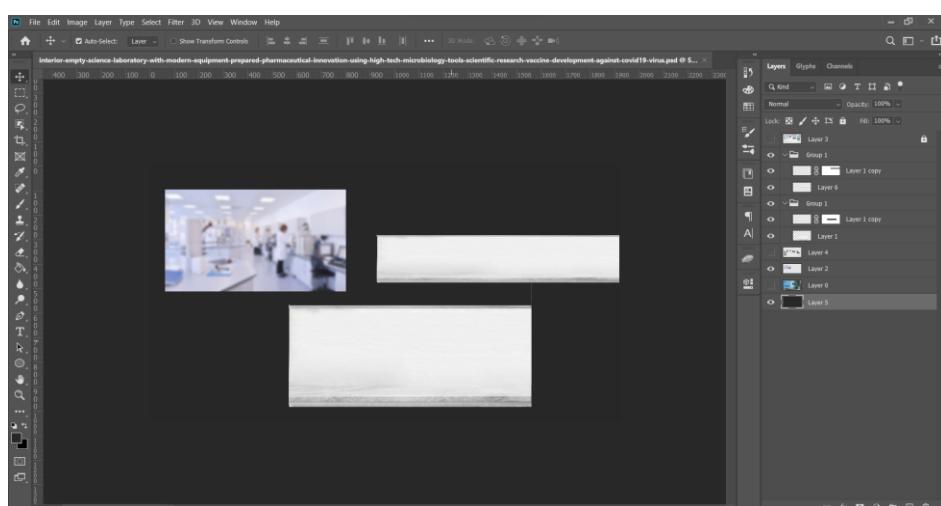
3.2.6 Scene 5 – Vật liệu và hóa chất

Thông tin về Scene

Phân cảnh 5 là khu vực để người dùng tìm hiểu khái niệm, công dụng các vật liệu và hóa chất được sử dụng trong thí nghiệm “Định lượng Chlorophyll bằng phương pháp đo quang phổ”.

Yêu cầu đặt ra:

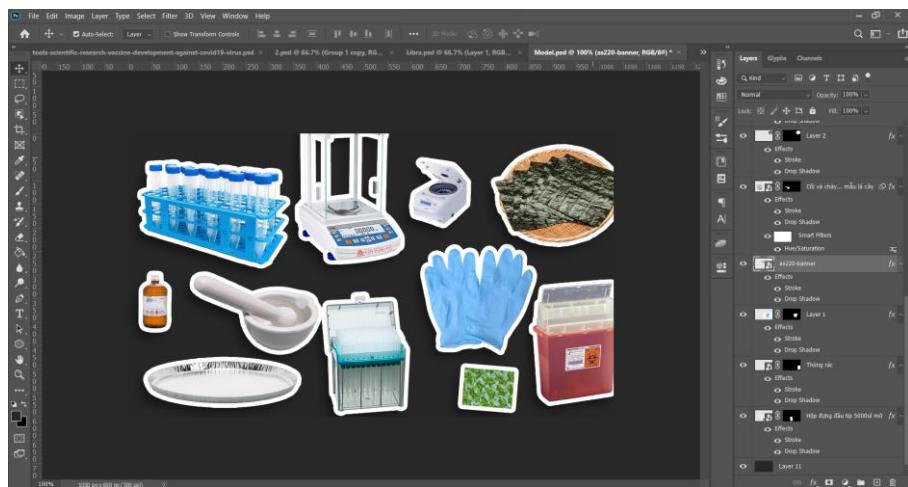
- Đầy đủ các hình ảnh vật liệu và hóa chất của thí nghiệm
- Thể hiện trạng thái Chưa xem/Đã xem của từng vật liệu và hóa chất
- Có giọng đọc AI



Hình 3.35. Các Components UI của Scene

Quy trình xử lý hình ảnh

Hình nền phía sau của Scene được xử lí bằng Adobe Photoshop, sử dụng phương pháp Gaussian Blur (Thuật toán làm mờ đồ họa theo hàm Gaussian) để làm mờ ảnh nền. Phần hình ảnh bàn thí nghiệm sẽ dùng Transform tool (Công cụ thay đổi hình dạng) để điều chỉnh Scale (phép biến hình) sao cho vừa với ảnh nền để tăng độ chân thật của ảnh nền thí nghiệm (*Hình 3.35*). Bên cạnh đó, các hình ảnh đồ họa của vật liệu, hóa chất sẽ cho thêm hiệu ứng viền từ Fx tool (Công cụ kĩ xảo) và đố phóng để làm nổi bật các đối tượng (*Hình 3.36*). Các đối tượng hình sẽ được kết xuất ở định dạng PNG thay vì JPG thông dụng. Vì định dạng PNG hỗ trợ các đối tượng đồ họa trong suốt, tạo cảm giác chân thật hơn.

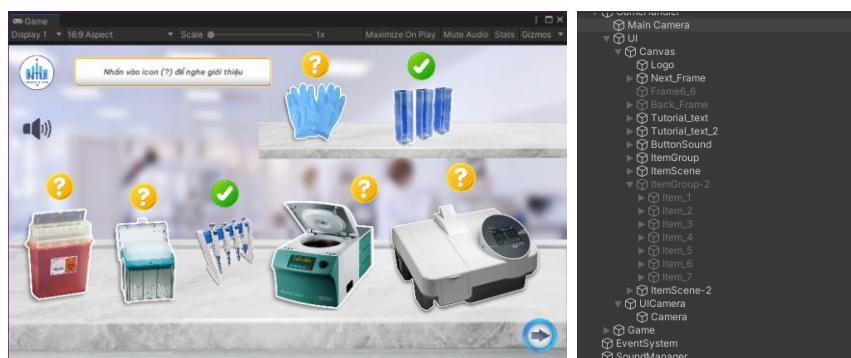


Hình 3.36. Các hình ảnh vật liệu sau khi được xử lí trên Photoshop

Quy trình xử lí Unity

Ý tưởng: Theo kịch bản, các vật liệu và hóa chất cần được giới thiệu với số lượng là 14 đối tượng. Do vậy, để tối ưu luồng làm, cần tạo một template GameObject với mỗi vật liệu sẽ có cấu trúc GameObject giống nhau nhưng khác nhau ở phần thông tin: nội dung và tên gọi. Vì vậy, template GameObject sẽ bao gồm những thành phần như sau:

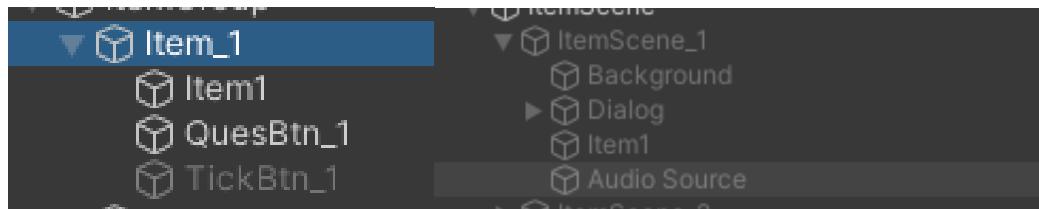
- QuesBtn/TickBtn: Hai biểu tượng đánh dấu trạng thái Chưa xem/Đã xem



Hình 3.37. Giao diện đồ họa của phân cảnh 5 (Bên trái) và cây phả hệ (Bên phải)

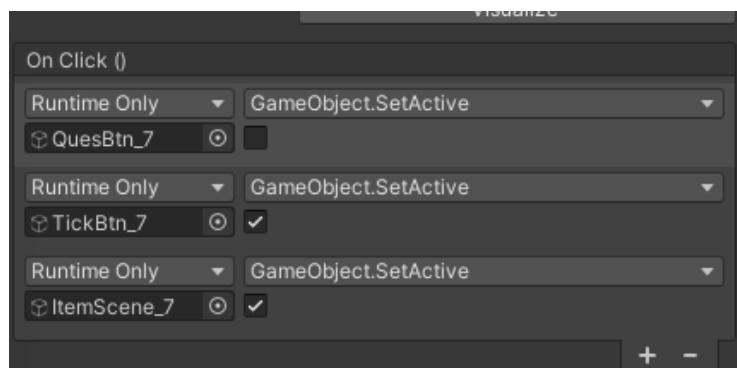
- Background: Hình ảnh đồ họa có màu đen mờ phủ lên những vật thể

- Dialog: Bao gồm các thông tin của đối tượng (“Thoát khỏi màn hình”)
- Item: Đối tượng đồ họa tượng trưng cho hình ảnh của vật liệu
- Audio Source: Âm thanh giọng đọc của thông tin đối tượng.



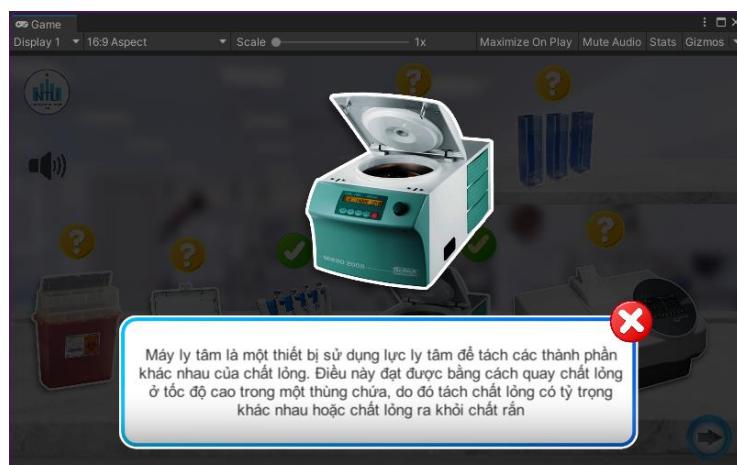
Hình 3.38. Template GameObject tiêu chuẩn

Ở mỗi template GameObject sẽ có thành phần QuesBtn/TickBtn. Để cài đặt sự kiện đánh dấu trạng thái Chưa xem/Đã xem, sử dụng giao diện kéo-thả và cài đặt các đối tượng vào OnClick() của Button “QuesBtn/TickBtn” (Hình 3.39).



Hình 3.39. Cài đặt sự kiện OnClick() cho Button

QuestBtn mặc định có giá trị False ở hàm SetActive(). Đồng thời, khởi tạo giá trị True cho Button “TickBtn” và GameObject “ItemScene”. ItemScene là GameObject tập hợp các GameObject con hiển thị các thông tin như hình ảnh và thông tin chi tiết của đối tượng (Hình 3.40).



Hình 3.40. Template cho vật liệu khi được nhấp vào

Tạo Animation tùy chỉnh cho ItemScene

Bên cạnh thư viện LeanTweener, sử dụng các hàm tùy chỉnh hoạt ảnh cũng là một cách tối ưu đối với các đối tượng đặc thù. C# Script dưới đây dùng để tạo một thư viện animation mới:

```
public class DialogItemSceneAnimation : MonoBehaviour
{
    public CanvasGroup item;
    public Transform box;
    public CanvasGroup background;
    private void OnEnable()
    {
        item.alpha = 0;
        item.LeanAlpha(1, 0.2f);

        background.alpha = 0;
        background.LeanAlpha(1, 0.5f);

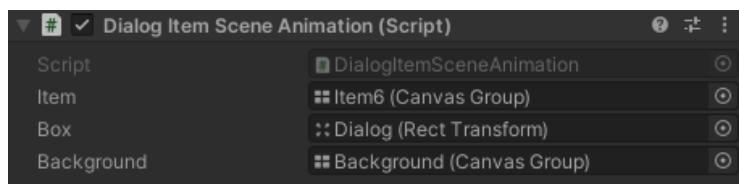
        box.localPosition = new Vector2(0, -Screen.height);
        box.LeanMoveLocalY(-103, 0.5f).setEaseOutExpo().delay = 0.1f;
    }
}
```

OnEnable() được hoạt động thực thi Animation của đối tượng với 2 thuộc tính: Position (phép biến đổi vị trí) và Alpha (Độ đặc của đối tượng). CloseDialog cũng hoạt động tương tự nhưng dành để khởi động animation ở trạng thái Tắt. Các biến khai báo mở sẽ được gán từ trong giao diện Unity.

```
public void CloseDialog()
{
    item.LeanAlpha(0, 0.2f);
    background.LeanAlpha(0, 0.5f);
    box.LeanMoveLocalY(-Screen.height, 0.5f).setEaseInExpo().setOnComplete(OnComplete);
}

void OnComplete()
{
    GameObject.SetActive(false);
}
```

Các đối tượng được gán vào các biến đã được khai báo từ trước. Trong đó, Item là hình ảnh đối tượng. Box là Thông tin chi tiết của đối tượng và Background là ảnh nền của đối tượng (*Hình 3.41*).



Hình 3.41. Khai báo các đối tượng vào C# Script

3.2.7 Scene 6 – Thực hành mô phỏng “Chuẩn bị vật liệu”

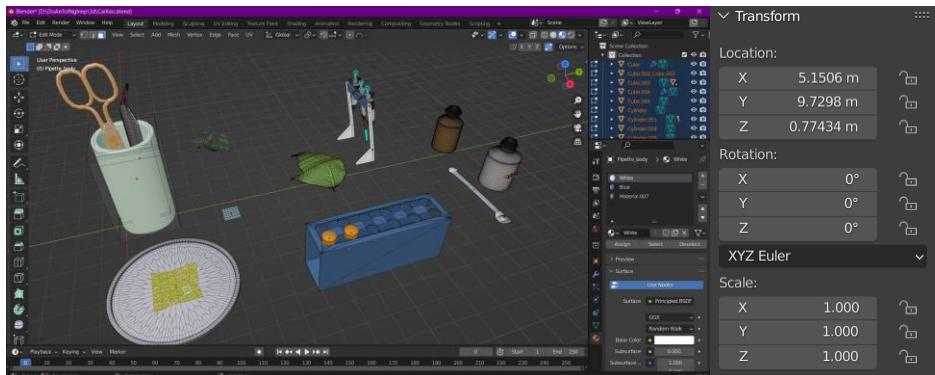
Thông tin về Scene

Phân cảnh 6 là khu vực mà người dùng có thể thực thi tương tác các bước mô phỏng của bài thí nghiệm. Người dùng được tương tác với các GameObject thông qua các nút tương tác, từ đó giúp người dùng hiểu hơn về các quy trình thực hành thí nghiệm.

Yêu cầu đặt ra:

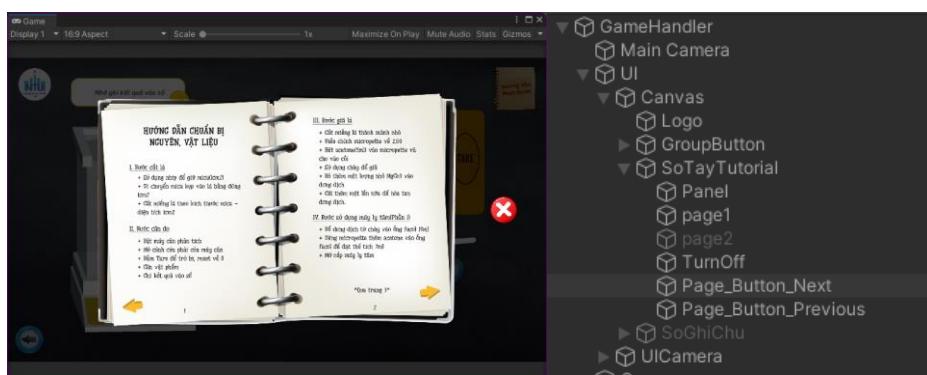
- Có số tay hướng dẫn người dùng
- Model của các dụng cụ thí nghiệm bám sát với hình ảnh đời thật
- Giao diện tương tác thực hành dễ sử dụng
- Các bước thực hành bám sát với kịch bản thí nghiệm

Quy trình xử lí Model



Hình 3.42. Giao diện xử lí model (Bên trái) và Bảng thuộc tính của 3D object (Bên phải)

Các vật liệu được modeling và sẵn sàng phục vụ cho assets của Unity. Lưu ý rằng, các đối tượng 3D cần được điều chỉnh các thông số Scale và Rotation về tỉ lệ 1:1 và 0 độ để tránh trường hợp lỗi data khi nhập vào Unity assets. Một số dụng cụ như kéo, nhíp có độ bám sát nguyên tắc ở mức tương đối vì độ đa dạng của chúng trong các phòng thí nghiệm. Sau khi đã tham khảo và hỏi đáp giảng viên phụ trách bài thí nghiệm, với những dụng cụ mang tính quyết định như Micropette, ống falcon,... đều đạt chuẩn về mức độ nhận dạng.



Hình 3.43. Sổ hướng dẫn trong Game và cây phả hệ SoTayTutorial

Quy trình xử lí Unity

Giao diện sổ tay hướng dẫn

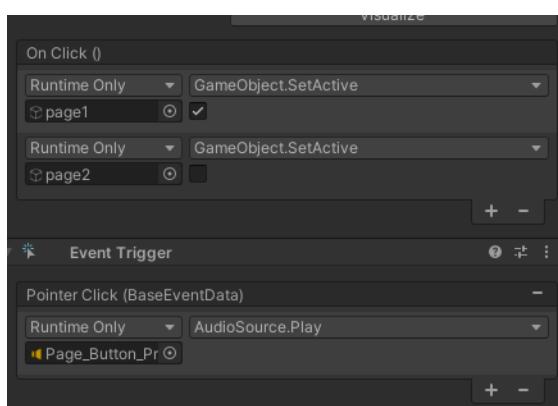
Sổ tay hướng dẫn được dùng để xem hướng dẫn các bước cần thực hiện khi người dùng cần xem lại quy trình thực hiện. Để cho phần Sổ tay hướng dẫn luôn nằm phía trên các phân cảnh (Hình 3.43), khởi tạo GameObject SoTayTutorial vào khu vực của UI. Khu vực hiển thị của UI chỉ hiển thị thông báo, nút tương tác không có mối liên hệ trực

tiếp đến với phân cảnh hiện tại.

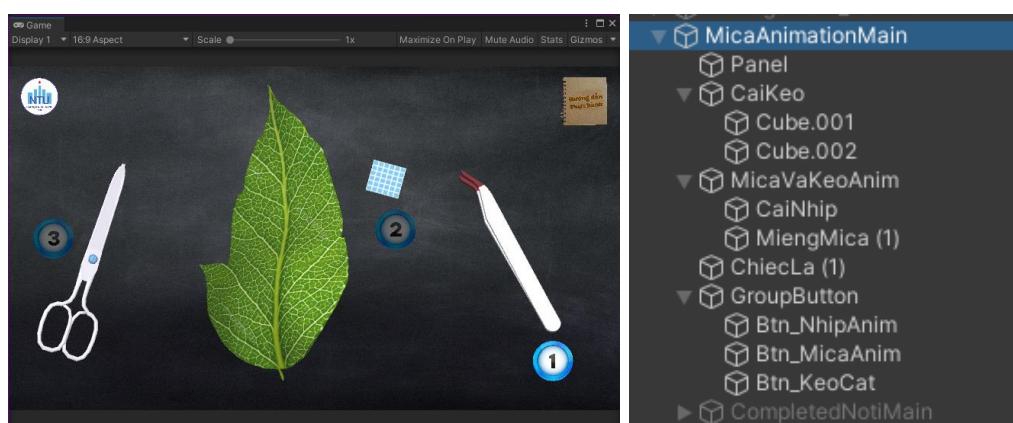
Các thành phần của SoTayTutorial bao gồm:

- Panel: Hình nền đen mờ nằm phía sau các đối tượng hiển thị
- Page: Đối tượng hình ảnh tượng trưng cho các trang hướng dẫn của Sổ tay hướng dẫn
- TurnOff: Nút tương tác dùng để thoát khỏi giao diện SoTayTutorial
- Page_Button: Các Button GameObject có chức năng chuyển trang sách

Ở hai nút tương tác “*Qua Trang*” hoặc “*Quay lại Trang trước*” sẽ được cài đặt các sự kiện OnClick() (Hình 3.44), để đảm bảo không bị xung đột các trang và sẽ phát âm thanh “*Tiếng lật sách*” khi nhấn vào.



Hình 3.44. Cài đặt OnClick() cho hai nút tương tác của SoTayTutorial

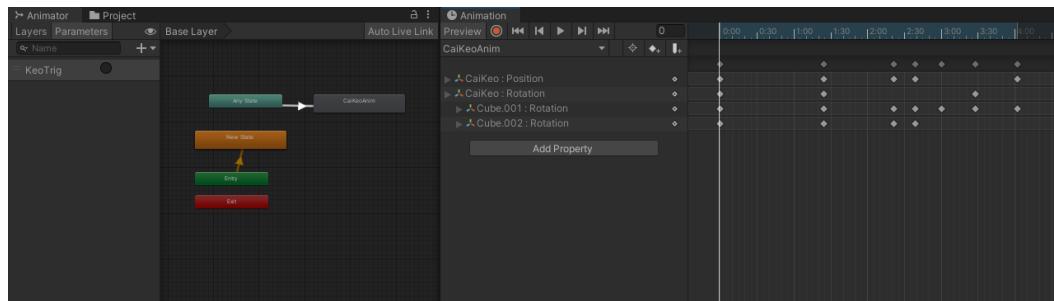


Hình 3.45. Giao diện “*Cắt Lá Theo Miếng Mica*” (Bên trái) và cây phả hệ (Bên phải)

Chuẩn bị cho bước “*Cắt Lá Theo Miếng Mica*”

Theo kịch bản, bước “*Cắt Lá*” được thực hiện theo thứ tự: 1. Cắt lá với diện tích chuẩn 1cm² (bằng giấy hoặc mica) và kẹp miếng mica vào lá. 2. Cắt một phần lá/rong khô bằng đúng 1cm² của miếng mica. Các vật thể 3D được nhập vào môi trường của Unity với việc điều chỉnh lại các thông số về Position, Scale sao cho phù hợp với Viewport Game được xuất từ Camera Game (Hình 3.45).

Đến với phần animation của vật thể, bằng việc sử dụng công cụ Animation và Animator của Unity, các animation của vật thể 3D được tạo ra với dữ liệu animation được quản lý trong thư mục “Animation”. Mỗi animator sẽ chứa một hoặc nhiều Controller và kích hoạt bằng cách gán biến cho đối tượng Controller như hình dưới là biến “KeoTrig”. Ở khu vực Animation cho phép người dùng điều chỉnh các thuộc tính Position, Rotation, Scale của đối tượng 3D chỉ định và thay đổi theo thuộc tính theo từng mốc Keyframe (Hình 3.46).



Hình 3.46. Timeline Animation của đối tượng Kéo

Mỗi GameObject sẽ có dữ liệu animation riêng và chỉ được thực thi khi nhận dữ liệu Trigger Parameters bằng sự kiện OnClick() của Button. Đoạn mã dưới đây được sử dụng để kích hoạt Trigger Parameters và khởi động animation của GameObject:

```
private int checkTrig = 0;
public Animator AnimatorMica;
public Animator AnimatorNhip;
public Animator AnimatorKeoCat;
public GameObject FinishNoti;
```

Khởi tạo các đối tượng có Components Animator và biến kiểu dữ liệu Integer “checkTrig”. Các đối tượng sẽ được khai báo thông qua giao diện kéo-thả trong Unity.

```
public void MoveNhip()
{
    if (checkTrig == 0)
    {
        checkTrig = 1;
        AnimatorNhip.SetTrigger("NhipTrig");
    }
}
```

Ví dụ, để gán biến kích hoạt Trigger Parameters của GameObject “Nhip”, sử dụng cú pháp như đoạn mã dưới:

⇒ *Đối_tượng_animator_đã_khai_báo.SetTrigger("Biến_khởi_khởi_tạo_tù_Animator")*

Biến “checkTrig” sẽ dùng như dữ liệu kiểm tra liệu animation của đối tượng đã được thực hiện hay chưa. CheckTrig có thể là biến kiểu dữ liệu Integer hoặc Bool, tùy vào độ phức tạp của khâu quản lý animation. Tương tự như các đối tượng 3D khác, các biến Trigger Parameter sẽ được thực thi thông qua đoạn mã dưới:

```
public void MoveMica()
{
    if (checkTrig == 1)
```

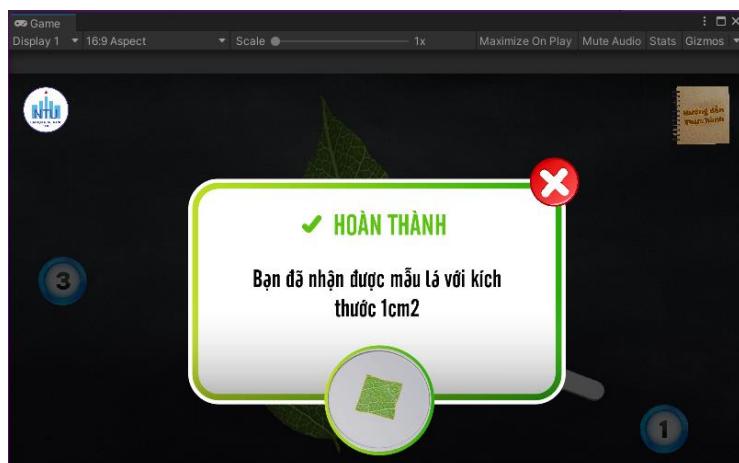
```

{
    checkTrig = 2;
    AnimatorMica.SetTrigger("MicaTrig");
}
public void CutKeo()
{
    if (checkTrig == 2)
    {
        checkTrig = 3;
        AnimatorKeoCat.SetTrigger("KeoTrig");
    }

    if(checkTrig == 3)
    {
        Invoke("CompleteStep1",4.5f);
    }
}
public void CompleteStep1()
{
    if (checkTrig == 3)
    {
        FinishNoti.SetActive(true);
    }
}

```

Để tạo OnClick() cho sự kiện Trigger Parameter, các Button tương tác được liên kết với GameObject tương ứng trong cho C# Script dùng để quản lý animation và từ đó, gán các hàm tương ứng vào OnClick() của các Button (*Hình 3.47*). Sau khi các hàm thực thi animation đã được đánh dấu hoàn thành, CompleteStep1() sẽ được thực thi và hiển thị thông báo hoàn thành bước 1.



Hình 3.47. Thông báo hoàn thành (Bên trái) và Gán sự kiện OnClick() (Bên phải)

Chuẩn bị bước “Đo lá bằng máy cân phân tích”

Theo kịch bản, bước “*Đo lá bằng máy cân phân tích*” được thực hiện theo thứ tự:

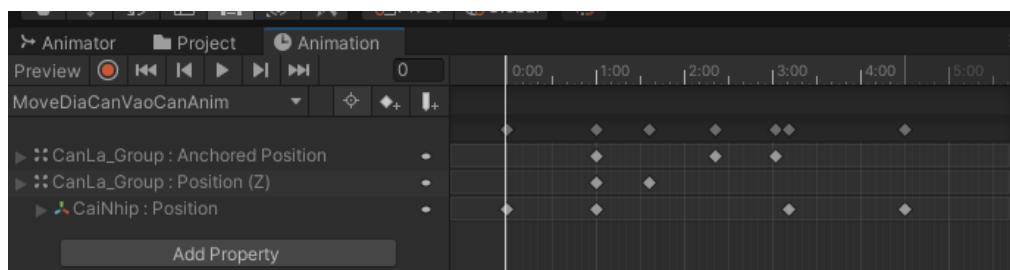
1. Mở nắp của máy cân đo phân tích;
2. Nhấn nút “Tare” để số đo về 0;
3. Dùng nhíp để gấp vật mẫu và bỏ vào máy;
4. Tiến hành đo bằng cách thao tác trên màn hình máy cân phân tích



Hình 3.48. Giao diện tương tác của “Đo lá bằng máy cân phân tích” (Bên trái) và Giao diện màn hình của máy cân phân tích (Bên phải)

Tương tự như phần thực hành tương tác của bước “*Cắt Lá Theo Miếng Mica*”, để giúp người dùng tương tác với các đối tượng 3D trên màn hình, các đối tượng cần được khởi tạo dữ liệu animation từ khu vực Animation và Animator (*Hình 3.49*). Khu vực Animation sẽ giúp lập trình viên điều khiển luồng dữ liệu animation và kích hoạt chúng bằng Trigger Parameters.

Sau đó, sử dụng giao diện lập trình kéo-thả của Unity, các nút tương tác sẽ được gán hàm thực thi chuyển dữ liệu vào Trigger Parameters và kích hoạt animation của đối tượng. Bên cạnh đó, phần giao diện màn hình của máy cân phân tích cũng có một số điểm lưu ý: Các dãy số của máy cân phân tích được thay đổi theo thứ tự với decimal là 8 chữ số “00.000.000”. Để khởi tạo được hiệu ứng chuyển động từ dãy số này sang dãy số khác bám sát với máy cân phân tích thực tế, đối tượng cần được lập trình như sau:



Hình 3.49. Khởi tạo dữ liệu animation cho máy cân phân tích

Sẽ có hai C# Script được khởi tạo bao gồm: “*TextCountNumber*” và “*TextCountNumberUpdater*”. *TextCountNumber* được sử dụng để tinh chỉnh các thông số cơ bản của đối tượng và thực thi các hàm tính toán. *TextCountNumberUpdater* sẽ được dùng để nhận dữ liệu đầu vào từ người dùng và chuyển dữ liệu sang *TextCountNumber*.

```
public class TextCountNumber : MonoBehaviour
{
    public TextMeshProUGUI Text;
    public int CountFPS = 30;
    public float Duration = 1f;
    public string NumberFormat = "D8";
    private int _value;
    public int Value
    {
        get
    }
```

```

    {
        return _value;
    }
    set
    {
        UpdateText(value);
        _value = value;
    }
}
private Coroutine CountingCoroutine;

private void Awake()
{
    Text = GetComponent<TextMeshProUGUI>();
}

```

TextCountNumber(): cho phép người dùng khai báo các thuộc tính cơ bản của đối tượng như định dạng dãy số, FPS (Khung hình trên giây), Duration (Thời gian chờ),...

```

private void UpdateText(int newValue)
{
    if (CountingCoroutine != null)
    {
        StopCoroutine(CountingCoroutine);
    }
    CountingCoroutine = StartCoroutine(CountText(newValue));
}
private IEnumerator CountText(int newValue)
{
    WaitForSeconds Wait = new WaitForSeconds(1f / CountFPS);
    int previousValue = _value;
    int stepAmount;

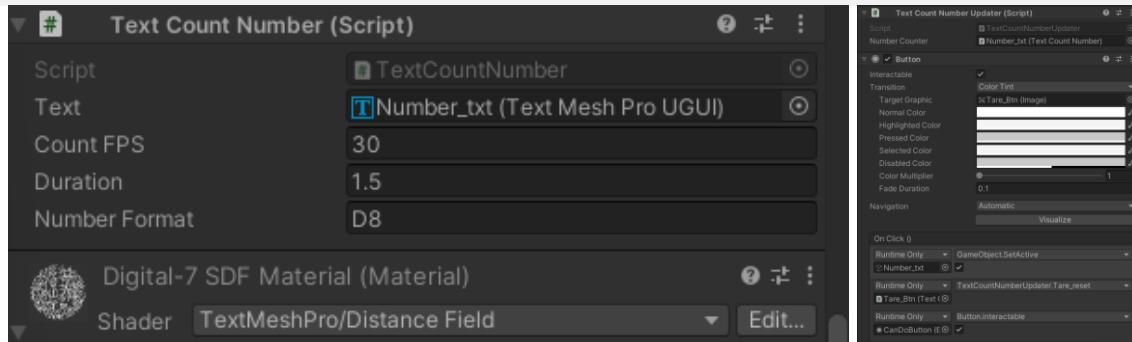
    if (newValue - previousValue < 0)
    {
        stepAmount = Mathf.FloorToInt((newValue - previousValue) / (CountFPS * Duration));
    }
    else
    {
        stepAmount = Mathf.CeilToInt((newValue - previousValue) / (CountFPS * Duration));
    }
    if (previousValue < newValue)
    {
        while (previousValue < newValue)
        {
            previousValue += stepAmount;
            if (previousValue > newValue)
            {
                previousValue = newValue;
            }
            Text.SetText(previousValue.ToString(NumberFormat));
            yield return Wait;
        }
    }
    else
    {
        while (previousValue > newValue)
        {
            previousValue -= stepAmount;
            if (previousValue < newValue)
            {
                previousValue = newValue;
            }
            Text.SetText(previousValue.ToString(NumberFormat));
            yield return Wait;
        }
    }
}

```

Interface CountText có chức năng để tạo thời gian của khoảng cách giữa 2 dãy số bằng đơn vị 1f / CountFPS (biến người dùng khai báo). Và sử dụng thư viện Mathf để thực thi làm tròn kết quả của biến “StepAmount”.

`TextCountNumberUpdater()`: dùng để update từ dãy số khởi đầu cho đến dãy số điểm đến. Các biến có thể được khai báo giá trị từ giao diện lập trình kéo-thả hoặc có thể gán giá trị từ trong phần C# Script cho các đối tượng.

```
public class TextCountNumberUpdater : MonoBehaviour
{
    public TextCountNumber NumberCounter;
    public void Start_Can()
    {
        int value = 397;
        NumberCounter.Value = value;
    }
    public void KhoiLuongDia_Mg()
    {
        int value = 21;
        NumberCounter.Value = value;
    }
    public void Delay_SetValue_ChangeToGram()
    {
        Invoke("KhoiLuongDia_Mg", 3.2f);
    }
    public void KhoiLuongDia_Gram()
    {
        int value = 19;
        NumberCounter.Value = (int)value;
    }
    public void Tare_reset()
    {
        int value = 00000000;
        NumberCounter.Value = value;
    }
}
```



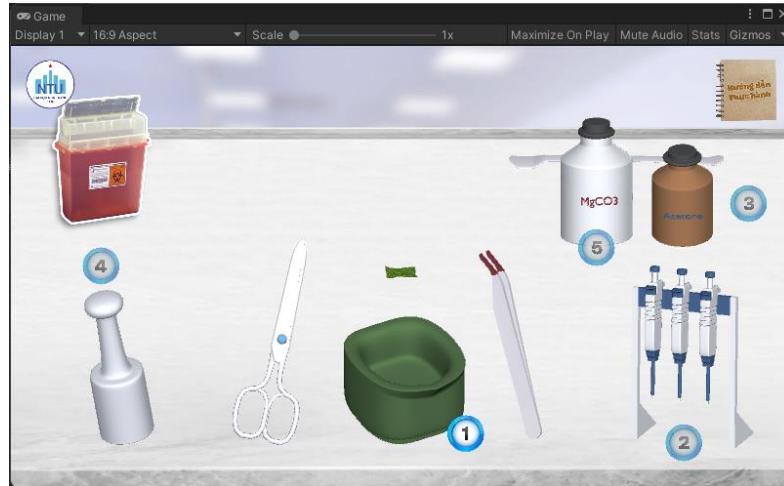
Hình 3.50. Khai báo đối tượng số vào TextCountNumber (Bên trái) và gán TextCountNumberUpdated vào Button tương tác (Bên phải)

TextCountNumberUpdated vào Button tương tác (Bên phải)

Sau khi hoàn thành phần lập trình, trở lại với Unity, các đối tượng Text GameObject cần được gán C# Script “TextCountNumber” và khai báo các giá trị của thông số Count FPS, Duration, Number Format. Nếu người dùng không khai báo, C# Script này sẽ tự động nhận giá trị mặc định từ phần mã lệnh (*Hình 3.50*).

Để các tương tác cho giao diện màn hình của máy cân phân tích, các đối tượng Button cần tương tác sẽ được gán những C# Script đã lập trình từ trước.

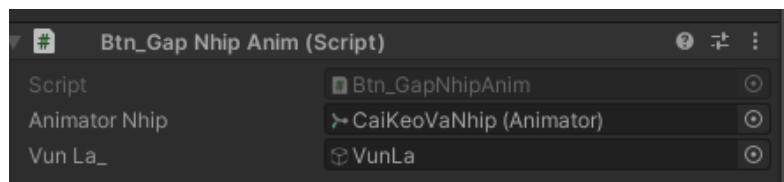
Chuẩn bị bước “Xử lý lá bằng hóa chất và chày cối”



Hình 3.51. Giao diện đồ họa “Xử lý lá bằng hóa chất và chày cối”

Theo kịch bản thí nghiệm, “*Xử lý lá bằng hóa chất và chày cối*” bao gồm các bước thực hiện sau: 1. Cắt mẫu vật thành những vụn lá có kích thước nhỏ; 2. Sử dụng Micropette và điều chỉnh thông số về 2.00ml; 3. Hút dung dịch acetone bằng Micropette và bơm vào cối; 4. Sử dụng chày để nghiền lá; 5. Đổ thêm dung dịch MgCO3 vào cối đạt thể tích chuẩn; 6. Nghiền lá thêm một lần nữa

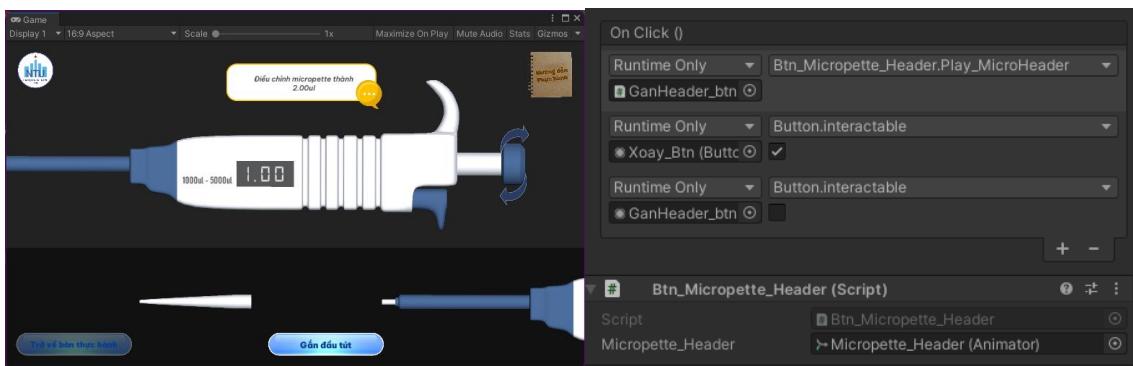
Bước “*Xử lý lá bằng hóa chất và chày cối*” vẫn sử dụng các công cụ Animation, Animator để tạo chuyển động tương tác với người dùng. Bằng việc sử dụng Trigger Parameter, và truyền dữ liệu bằng Button tương tác. Tương tự với những animation trước đây, C# Script được gán biến vào Button tương tác với người dùng. Theo đó là, khai báo các đối tượng 3D có liên quan (Hình 3.52):



Hình 3.52. Khai báo đối tượng Animator để kích hoạt Trigger

Trong khu vực của Micropette, khởi tạo một GameObject lớn và chỉ được hiển thị sau khi hoàn thành bước 1. “*Điều chỉnh Micropette*” sẽ có hai cửa sổ tương tác:

- Cửa sổ gắn đầu tút cho Micropette
- Xoay số ml của micropette thành 2.00ml



Hình 3.53. GameObject Micropette được giả lập như một Scene

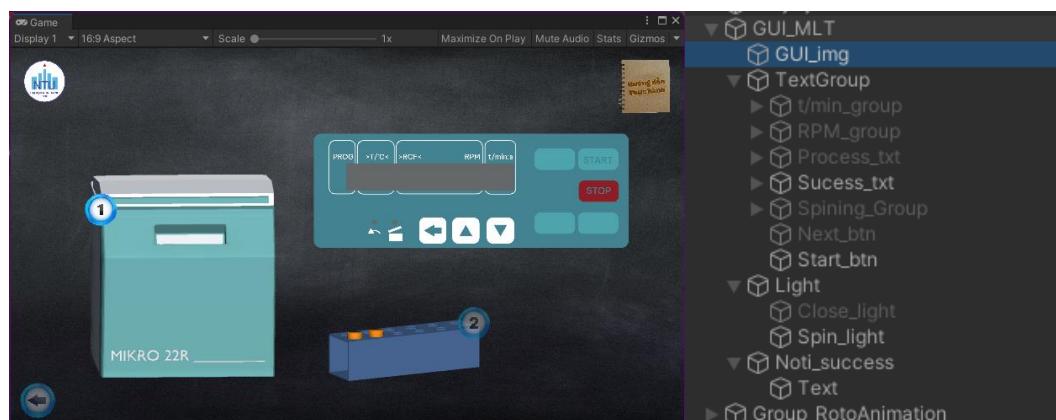
Ở cửa sổ “*Gắn đầu tút cho Micropette*”, khởi tạo C# Script để truyền dữ liệu vào Trigger Parameters của Animation. Trong đó, *Micropette_Header* là đối tượng Animator tương tác với dùng thông qua Button tương tác.

```
public class Btn_Micropette_Header : MonoBehaviour
{
    public Animator Micropette_Header;
    private int checkTrig = 0;
    public void Play_MicroHeader()
    {
        if (checkTrig == 0)
        {
            checkTrig = 1;
            Micropette_Header.SetTrigger("GanHeader");
        }
    }
}
```

Sau đó, khai báo đối tượng Animator cần thêm vào, tương tự như các Animator trước. C# Script được gán vào Button tương tác với người dùng và cài đặt sự kiện OnClick() với hàm Play_MicroHeader() (*Hình 3.53*).

Chuẩn bị bước “Sử dụng máy ly tâm cho vật mẫu”

Theo kịch bản, bước “*Sử dụng máy ly tâm cho vật mẫu*” sẽ thực hiện bao gồm các bước: 1. Mở nắp máy ly tâm; 2. Đặt hai ống falcon vào máy ly tâm; 3. Bật giao diện màn hình của máy ly tâm; 4. Điều chỉnh thông số và khởi động máy ly tâm; 5. Đặt ống falcon lại vào khay khi hoàn thành.



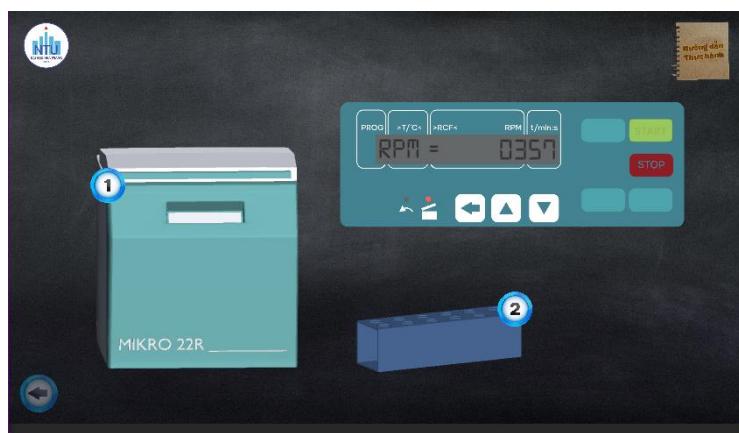
Hình 3.54. Giao diện đồ họa của “Sử dụng máy ly tâm cho vật mẫu” (Bên trái) và cây phả hệ (Bên phải)

Bên cạnh xử lý các nút tương tác để truyền dữ liệu animation vào máy ly tâm, còn một số vấn đề của màn hình máy ly tâm cần được xử lý. Trong đó, việc hiển thị màn hình trên dây số và điều chỉnh số vòng quay có thể tái sử dụng C# script TextNumberCounter đã lập trình từ trước để ứng dụng tương tự vào đối tượng.

Để tạo sự kiện tương tác cho người dùng với phần điều chỉnh t/min (thời gian xoay của máy ly tâm), đối tượng dây số được lập trình như sau:

```
public class tmin_script : MonoBehaviour
{
    private int textNumber;
    public Text TextObject;
    public GameObject Next_btn;
    public void minusOne()
    {
        textNumber = int.Parse(TextObject.text);
        if (TextObject != null && textNumber > 2)
        {
            --textNumber;
            TextObject.text = textNumber.ToString("D2");
            if(textNumber == 2)
            {
                Next_btn.SetActive(true);
            }
        }
    }
}
```

MinusOne() được dùng để tăng hoặc giảm giá trị thời gian của vòng xoay máy ly tâm và theo yêu cầu của bài thí nghiệm, người dùng cần điều chỉnh giá trị t/min = 2. Vì vậy, đoạn mã có câu điều kiện If để kiểm tra liệu người dùng đã đặt t/min đúng giá trị hay chưa. Khi đã thỏa mãn điều kiện, nút tương tác “RPM Điều chỉnh số vòng xoay” sẽ được hiển thị (*Hình 3.55*).



Hình 3.55. Hiệu chỉnh số vòng quay của máy ly tâm

Bên cạnh đó, để xử lý các thành phần ẩn và hiển thị GameObject của màn hình. Để thực hiện việc mô phỏng quá trình xử lý của máy ly tâm, bắt đầu với phần mã lệnh:

```
public class Timer_Txt_Spining : MonoBehaviour
{
    public GameObject Txt_success;
    public GameObject Txt_process;
    public GameObject Noti_sucess;
    public GameObject Process_txt;
    void Start()
```

```

    {
        Invoke("Process_txt_", 3f);
        Invoke("Spining_success", 10f);
    }
}

```

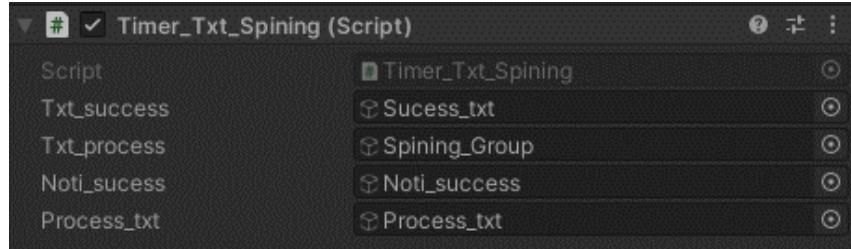
Start() sẽ tự thực thi khi được khởi động GameObject cha “*Spining_Group*” và thực hiện hai hàm Process_txt(), Spining_success() theo thời gian được cài đặt.

```

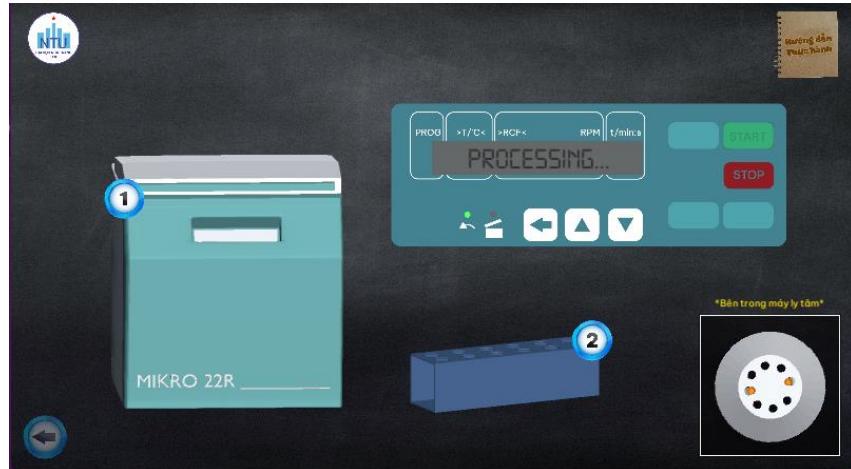
public void Spining_success()
{
    Txt_success.SetActive(true);
    Noti_sucess.SetActive(true);
    Process_txt.SetActive(false);
}
public void Process_txt_()
{
    Process_txt.SetActive(true);
    Txt_process.SetActive(false);
}
}

```

Ở phần giao diện Unity, khai báo các đối tượng GameObject được sử dụng để hiển thị trong quá trình máy ly tâm đang xoay. Khi đối tượng “*Spining_Group*” ở trạng thái “Bật”, C# Script này sẽ được thực thi tự động (*Hình 3.56*).



Hình 3.56 Khai báo các đối tượng cần hiển thị ở màn hình máy ly tâm



Hình 3.57. Hình ảnh máy ly tâm hoạt động trong Unity

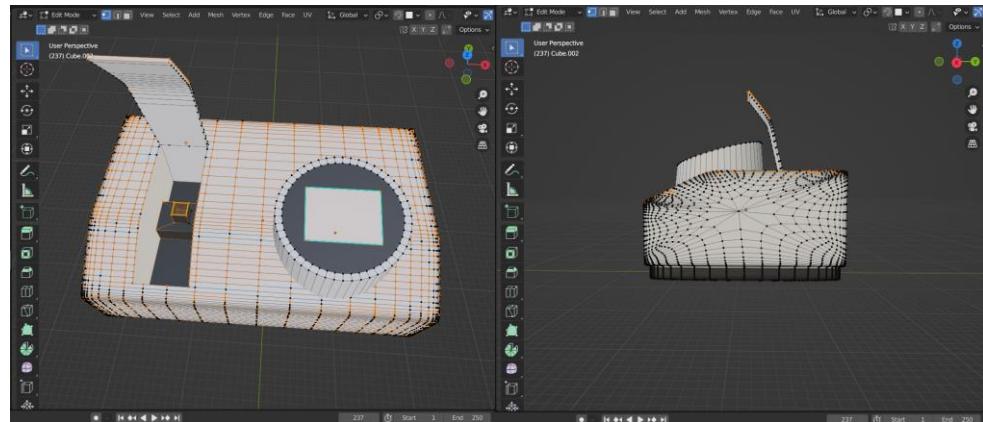
3.2.8 Scene 7 – Thực hành mô phỏng “Sử dụng máy đo quang phổ”

Thông tin về Scene

Scene 7 là khu vực thực hành sử dụng máy đo quang phổ. Người dùng được tương tác với máy đo quang phổ với các yêu cầu của thí nghiệm. Sau khi hoàn thành phần thực hành, người dùng cần hoàn thành bài báo cáo của thí nghiệm thông qua “*Sổ báo cáo*”.

Quy trình xử lý Model

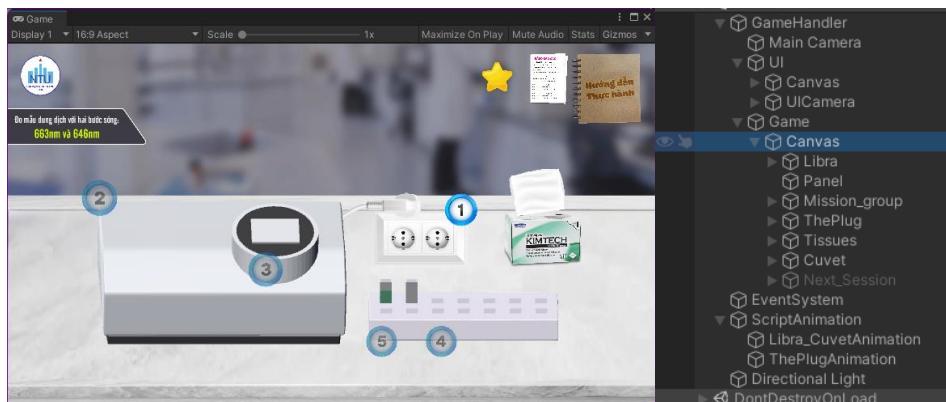
Mẫu máy đo quang phổ của Viện Công nghệ Sinh học và Môi trường tại Trường Đại học Nha Trang sử dụng có tên là “*Libra UV VIS S50*”. Các hình ảnh của mẫu máy đã được chụp lại và modeling theo tỉ lệ 1:1. Vật thể 3D sau khi modeling có chuyển động đóng/mở nắp máy (*Hình 3.58*).Thêm vào đó, màn hình giao diện máy được tách riêng với phần máy và được thiết kế thành giao diện riêng để thuận tiện trong quá trình mô phỏng của Unity.



Hình 3.58. Mặt trước và mặt sau của máy đo quang phổ trong Blender

Quy trình xử lý Unity

Ý tưởng: Quản lý toàn bộ các biến Trigger của Animation bằng một C# Script và khai báo đối tượng của vật thể, kể cả màn hình giao diện của máy đo quang phổ. Sau khi hoàn thành, ở phần UI sẽ hiển thị GameObject “Số báo cáo” để người dùng nhập các kết quả đo của thí nghiệm.



Hình 3.59. Giao diện “Sử dụng máy đo quang phổ” (Bên trái) và cây phả hệ (Bên phải)

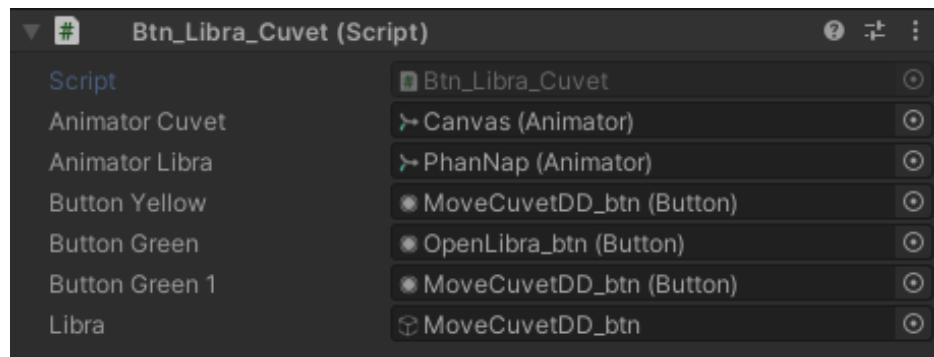
Theo kịch bản “*Sử dụng máy đo quang phổ*”, người dùng sẽ thực hiện các bước như sau: 1. Kết nối nguồn điện với máy quang phổ; 2. Mở nắp và khởi động máy quang phổ; 3. Đặt mẫu cuvet blank vào trong máy và tính toán; 4. Đặt mẫu cuvet dung dịch vào trong máy và tính toán; 5. Nhập kết quả vào sổ báo cáo

Chuẩn bị C# Script quản lý dữ liệu animation của vật thể 3D

Đầu tiên, khởi tạo C# Script để quản lý các Animation và Animator của phân cảnh. Khởi tạo các đối tượng Animator và được khai báo từ giao diện kéo-thả của Unity. Bắt đầu thiết lập các hàm truyền dữ liệu vào Trigger Parameters.

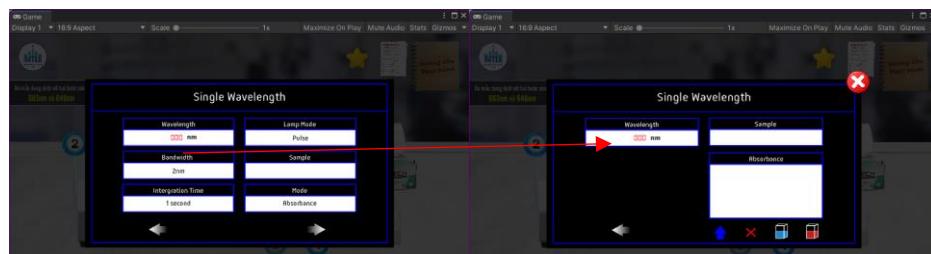
```
public class Btn.Libra_Cuvet : MonoBehaviour
{
    public Animator AnimatorCuvet;
    public Animator AnimatorLibra;
    public Button ButtonYellow;
    public Button ButtonGreen;
    public Button ButtonGreen1;
    public GameObject Libra;
    private int checkTrig = 0;
    private int checkMove = 0;
    public void Libra_btn()
    {
        if (checkTrig == 0)
        {
            checkTrig = 1;
            AnimatorLibra.SetTrigger("TrigOpen");
        }
        else if (checkTrig == 1)
        {
            checkTrig = 0;
            AnimatorLibra.SetTrigger("TrigClose");
        }
    }
    public void Cuvet_btn_Bank()
    {
        if (checkTrig == 1 && checkMove == 0)
        {
            checkMove = 2;
            AnimatorCuvet.SetTrigger("Blank_trig");
        }
        else if (checkMove == 3)
        {
            checkMove = 4;
            AnimatorCuvet.SetTrigger("Blank_trig_2");
        }
    }
    public void Cuvet_btn_DD()
    {
        if (checkMove == 2)
        {
            checkMove = 3;
            AnimatorCuvet.SetTrigger("DungDich_trig");
        }
        else if (checkMove == 3)
        {
            AnimatorCuvet.SetTrigger("DungDich_trig");
        }
        else if (checkMove == 4)
        {
            AnimatorCuvet.SetTrigger("DungDich_trig");
        }
    }
}
```

Các hàm sẽ được kiểm tra bởi biến “*checkMove*” với mục đích là gán giá trị theo kiểu dữ liệu Integer (1,2,3,4...) tương ứng cho các bước thực hiện theo thứ tự. Ở giao diện Unity, khai báo các đối tượng Animator tương ứng với các đối tượng trong C# Script (*Hình 3.60*):



Hình 3.60. Khai báo các đối tượng Animator vào C# script quản lý

Cài đặt giao diện màn hình của máy đo quang phổ



Hình 3.61. Các trang màn hình của máy đo quang phổ

Giao diện màn hình của máy đo quang phổ có hai trang chính: Trang điều chỉnh wavelength và trang tính toán cuvet (*Hình 3.61*). Ở ô trống Wavelength sẽ có một InputField để nhập số theo kiểu Integer Number và được giới hạn ở 3 số do mức sóng theo yêu cầu của thí nghiệm chỉ giao động ở 3 chữ số (645nm/663nm). Bên cạnh đó, khi dữ liệu ở wavelength InputField được nhập, wavelength bên trang tính toán cuvet sẽ được cập nhật theo thời gian thực. Ở trang tính toán cuvet có 2 Button tương tác: biểu tượng màu xanh và màu đỏ tượng trưng cho nút đo mẫu blank hoặc mẫu dung dịch. Hai Button có câu điều kiện kiểm tra xem liệu người dùng đã nhập đúng dãy sóng (wavelength) chưa. Nếu chưa thỏa mãn điều kiện, phần Absorance sẽ không hiển thị.

```
public class Libra_Caculate_script : MonoBehaviour
{
    public InputField wavelength_input;
    public Text Result_Absorbance;
    public Text Result_Wavelength;
    public Text Sample_txt;
    void Start()
    {
        wavelength_input.text = "000";
    }
    void Update()
    {
        Result_Wavelength.text = wavelength_input.text.ToString();
    }
}
```

Update() có chức năng truyền dữ liệu thời gian thực từ đối tượng Text này sang đối tượng Text khác.

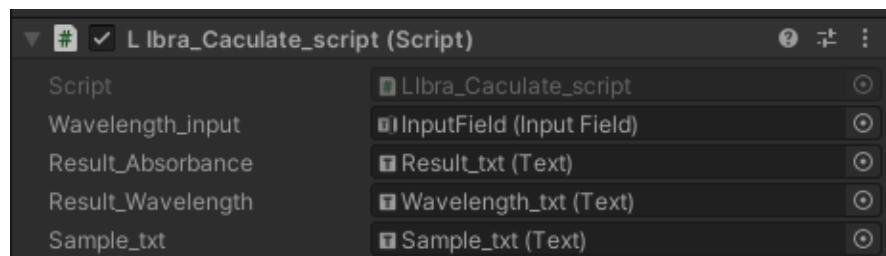
```
public void Blank_Cal()
{
    if (wavelength_input.text == "645" || wavelength_input.text == "663")
    {
        Result_Absorbance.text = "0.000";
    }
}
```

```

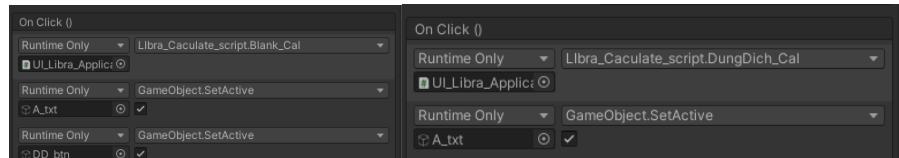
        Sample_txt.text = "Reference";
    }
}
public void DungDich_Cal()
{
    if(wavelength_input.text == "645" || wavelength_input.text == "663")
    {
        float nums = Random.Range(0.2f, 0.9f);
        double round_nums = System.Math.Round(nums, 3);
        Result_Absorbance.text = round_nums.ToString();
        Sample_txt.text = "1";
    }
}

```

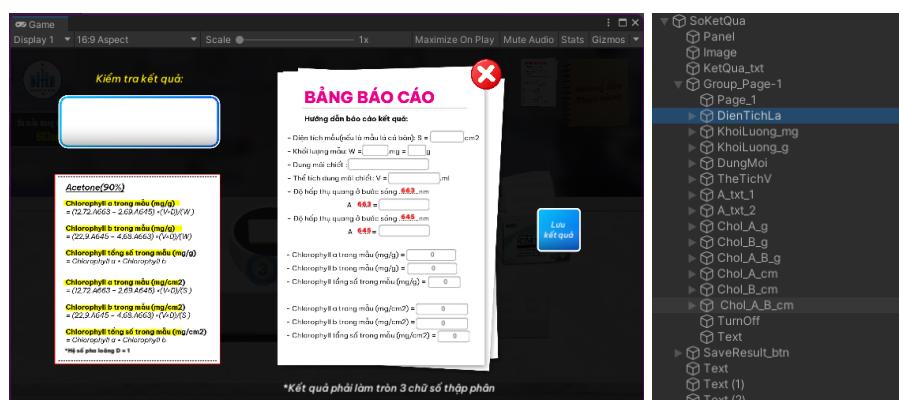
Blank_Cal() sẽ dùng để kiểm tra liệu người dùng đã nhập đúng số liệu dãy sóng và đặt lại dữ liệu bằng 0 nhập đúng dãy sóng. Hàm DungDich_Cal() dùng để kiểm tra liệu wavelength InputField có nhập đúng số bước sóng theo yêu cầu của thí nghiệm. Khi thỏa mãn điều kiện biến nums sẽ được khởi tạo để chạy ngẫu nhiên từ giá trị 0.2 – 0.9. Sau đó, nums sẽ được làm tròn thành 3 số thập phân. Cuối cùng là sẽ hiển thị giá trị cuối vào màn hình.



Hình 3.62. Khai báo đối tượng GameObject vào C# Script



Hình 3.63. Cài đặt C# Script cho các Button thực thi



Hình 3.64 Giao diện Sổ báo cáo (Bên trái) và cây phả hệ (Bên phải)

Cuối cùng, khai báo các đối tượng cần được lập trình vào khu vực quản lý của C# Script (*Hình 3.62*). Bên cạnh đó, các Button tương tác sẽ được cài đặt C# Script và gán các hàm tính toán tương ứng (*Hình 3.63*).

Cài đặt Sổ báo cáo cho người dùng

Sau khi thực hiện các bước tính toán của máy đo quang phổ, số báo cáo được hiển thị và cho phép người dùng nhập liệu các thông tin tính toán. Các khoảng trống trong hình là GameObject có thuộc tính InputField (*Hình 3.64*).

Số báo cáo sẽ quản lý các thông tin nhập liệu người dùng và kiểm tra kết quả nhập liệu bằng các hàm kiểm tra được lập trình:

```
public class BaoCaoKQ_script : MonoBehaviour
{
    bool CheckKQ_1, CheckKQ_2, CheckKQ_3, CheckKQ_4, CheckKQ_5, CheckKQ_6;
    public GameObject NextSession;
    public Text TheTichV_txt;
    public Text DienTichS_txt;
    public Text KhoiLuongMauW_txt;
    public Text A_663_txt;
    public Text A_645_txt;
    public Text Chol_A_txt;
    public Text Chol_B_txt;
    public Text TongChol_AB_txt;
    public Text Chol_A_cm_txt;
    public Text Chol_B_cm_txt;
    public Text TongChol_AB_cm_txt;
    public Text Notifacation_;
    double KQ_Chol_B_r; KQ_Chol_A_r; KQ_Chol_B_cm_r; KQ_Chol_A_cm_r; A_663, A_645; TheTichV;
    double KQ_g; KQ_Chol_A; KQ_Chol_B; KQ_cm; KQ_Chol_A_cm; KQ_Chol_B_cm;
    double DienTichS; KhoiLuongMauW; Chol_A; Chol_B; TongChol_AB; Chol_A_cm; Chol_B_cm; TongChol_AB_cm
    int HeSoD = 1;
    public void CheckKQ()
    {
        A_663 = double.Parse(A_663_txt.text);
        A_645 = double.Parse(A_645_txt.text);
        TheTichV = double.Parse(TheTichV_txt.text);
        DienTichS = double.Parse(DienTichS_txt.text);
        KhoiLuongMauW = double.Parse(KhoiLuongMauW_txt.text);
        Chol_A = double.Parse(Chol_A_txt.text);
        Chol_B = double.Parse(Chol_B_txt.text);
        TongChol_AB = double.Parse(TongChol_AB_txt.text);
        Chol_A_cm = double.Parse(Chol_A_cm_txt.text);
        Chol_B_cm = double.Parse(Chol_B_cm_txt.text);
        TongChol_AB_cm = double.Parse(TongChol_AB_cm_txt.text);
        Tinh_CholA_g();
        Tinh_CholB_g();
        TongTinh_CholAB_g();
        Tinh_CholA_cm();
        Tinh_CholB_cm();
        TongTinh_CholAB_cm();
        if (CheckKQ_1 == true && CheckKQ_2 == true && CheckKQ_3 == true && CheckKQ_4 == true && CheckKQ_5 == true && CheckKQ_6 == true)
        {
            NextSession.SetActive(true);
            Notifacation_.text = "Toan bo ket qua da dung!";
        }
        else if(CheckKQ_1 == false)
        {
            Notifacation_.text = "Cholorophyll a (mg/g): Khong Dung!";
        }
        else if (CheckKQ_2 == false)
        {
            Notifacation_.text = "Cholorophyll b (mg/g): Khong Dung!";
        }
        else if (CheckKQ_3 == false)
        {
            Notifacation_.text = "Tong Cholorophyll A,B (mg/g): Khong Dung!";
        }
        else if (CheckKQ_4 == false)
        {
            Notifacation_.text = "Cholorophyll a (mg/cm2): Khong Dung!";
        }
        else if (CheckKQ_5 == false)
        {
            Notifacation_.text = "Cholorophyll b (mg/cm2): Khong Dung!";
        }
        else if (CheckKQ_6 == false)
        {
            Notifacation_.text = "He so D: Khong Dung!";
        }
    }
}
```

```

    {
        Notifacation_.text = "Tong Cholorophyll A,B (mg/cm2) Khong Dung!";
    }
}

```

CheckKQ() là một hàm tổng được thực thi khi người dùng Click vào Button “Lưu kết quả”. Dùng để thực thi chuyển đổi kiểu dữ liệu từ InputField vào C# Script, tính toán theo công thức của bài thí nghiệm, kiểm tra dữ liệu tính toán của người dùng đã khớp với kết quả của chương trình. Tùy vào trường hợp, thông báo kiểm tra kết quả sẽ đưa ra thông báo theo ngữ cảnh.

```

//Tinh cho Cholororyl A-B (mg/g)
private void Tinh_CholA_g()
{
    KQ_Chol_A = (12.72 * A_663 - 2.69 * A_645) * (TheTichV * HeSoD) / KhoiLuongMauW;
    KQ_Chol_A_r = Math.Round(KQ_Chol_A, 3);
    if (KQ_Chol_A_r == Chol_A)
    {
        CheckKQ_1 = true;
    }
    else
    {
        CheckKQ_1 = false;
    }
}
private void Tinh_CholB_g()
{
    KQ_Chol_B = (22.9 * A_645 - 4.68 * A_663) * (TheTichV * HeSoD) / KhoiLuongMauW;
    KQ_Chol_B_r = Math.Round(KQ_Chol_B, 3);
    if (KQ_Chol_B_r == Chol_B)
    {
        CheckKQ_2 = true;
    }
    else
    {
        CheckKQ_2 = false;
    }
}
private void TongTinh_CholAB_g()
{
    KQ_g = KQ_Chol_A_r + KQ_Chol_B_r;
    double KQ_Chol_Tong_r = Math.Round(KQ_g, 3);
    if (KQ_Chol_Tong_r == TongChol_AB)
    {
        CheckKQ_3 = true;
    }
    else
    {
        CheckKQ_3 = false;
    }
}
private void Tinh_CholA_cm()
{
    KQ_Chol_A_cm = (12.72 * A_663 - 2.69 * A_645) * (TheTichV * HeSoD) / DienTichS;
    KQ_Chol_A_cm_r = Math.Round(KQ_Chol_A_cm, 3);
    if (KQ_Chol_A_cm_r == Chol_A_cm)
    {
        CheckKQ_4 = true;
    }
    else
    {
        CheckKQ_4 = false;
    }
}
private void Tinh_CholB_cm()
{
    KQ_Chol_B_cm = (22.9 * A_645 - 4.68 * A_663) * (TheTichV * HeSoD) / DienTichS;
    KQ_Chol_B_cm_r = Math.Round(KQ_Chol_B_cm, 3);
    if (KQ_Chol_B_cm_r == Chol_B_cm)
    {
        CheckKQ_5 = true;
    }
    else

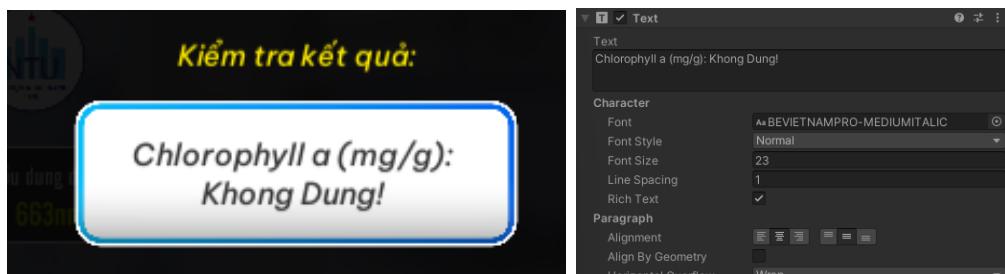
```

```

    {
        CheckKQ_5 = false;
    }
}
private void TongTinh_CholAB_cm()
{
    KQ_cm = KQ_Chol_A_cm_r + KQ_Chol_B_cm_r;
    double KQ_Chol_Tong_r_cm = Math.Round(KQ_cm, 3);
    if (KQ_Chol_Tong_r_cm == TongChol_AB_cm)
    {
        CheckKQ_6 = true;
    }
    else
    {
        CheckKQ_6 = false;
    }
}

```

Các công thức tính Chlorophyll của mẫu vật được lập trình tính toán dựa trên các công thức của bài thí nghiệm. Sau khi tính xong, các kết quả được làm tròn bằng hàm Math.Round() để trả về kết quả với 3 chữ số thập phân. Nếu kết quả không đúng hoặc thiếu, chúng sẽ hiển thị thông báo kết quả thất bại. Ví dụ: nếu tính kết quả Chlorophyll a không đúng, thông báo sẽ hiển thị như sau:



Hình 3.65. Thông báo lỗi (Bên trái) và đối tượng Text được truyền dữ liệu (Bên phải)

Scene 7 – Kiểm tra kiến thức

Thông tin về Scene

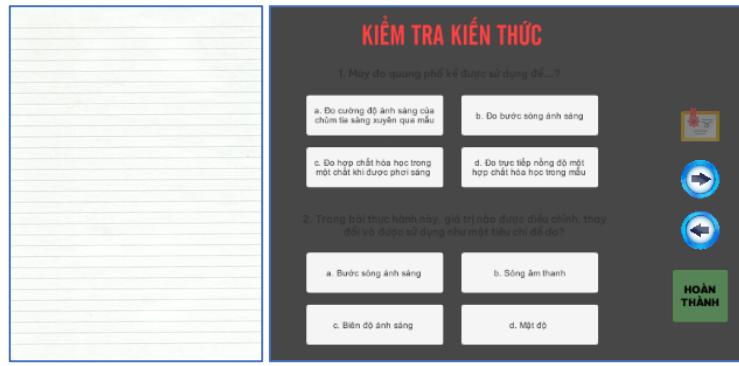
Kiểm tra kiến thức là phân cảnh để kiểm tra kiến thức người dùng thông qua dạng câu hỏi trắc nghiệm.

Yêu cầu đặt ra:

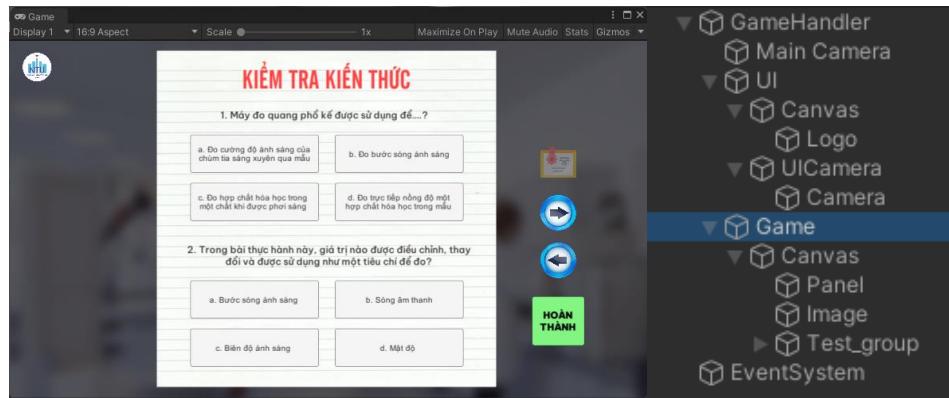
- Tính tổng điểm và đánh giá “Đạt” hay “Không Đạt”
- Sử dụng dạng câu hỏi Trắc nghiệm

Quy trình xử lý hình ảnh

Phân cảnh 7 sử dụng các đối tượng đồ họa: hình ảnh trang giấy định dạng JPEG và các biểu tượng cho Button GameObject. Các câu trả lời của câu hỏi được xây dựng trên Button GameObject và chứa Text. Tiêu đề của phân cảnh là dạng Text GameObject.



Hình 3.66. Components của phân cảnh được thiết kế bằng Adobe Illustrator
Quy trình xử lý Unity



Hình 3.67. Giao diện của Scene kiểm tra và cây phả hệ

Ý tưởng: Các câu hỏi được chứa trong GameObject cha “Test_group”. Mỗi câu trả lời đều khởi tạo ở dạng Button GameObject. Người dùng nhấn vào các Button, chọn đúng Button, các giá trị đã đăng ký ở OnClick() của các Button được thực thi và đặt trạng thái đúng cho câu trả lời.

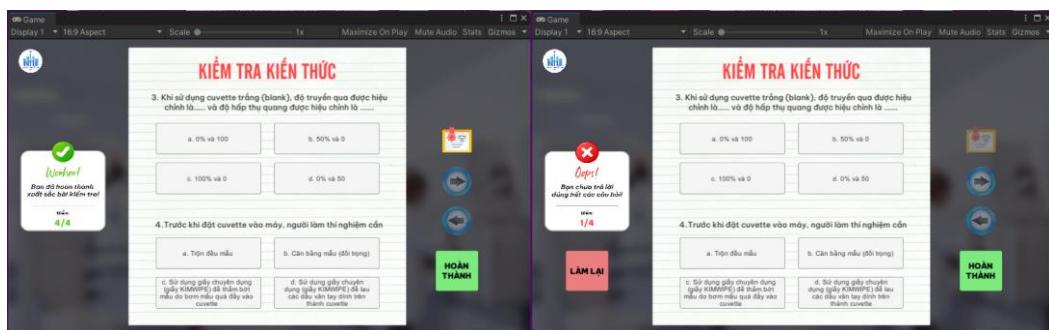
```
public class Test_Script : MonoBehaviour
{
    bool DapAn1 = false; DapAn2 = false; DapAn3 = false; DapAn4 = false;
    int Diem1 = 0; Diem2 = 0; Diem3 = 0; Diem4 = 0;
    public GameObject Reset_btn;
    public Button NhanBang_btn;
    public GameObject Thanhcong_noti;
    public GameObject ThatBai_noti;
    public Text Diem;
    public void Answer_1()
    {
        Diem1 = 1;
        DapAn1 = true;
    }
    public void Answer_2()
    {
        Diem2 = 1;
        DapAn2 = true;
    }
    public void Answer_3()
    {
        Diem3 = 1;
        DapAn3 = true;
    }
    public void Answer_4()
    {
        Diem4 = 1;
        DapAn4 = true;
    }
    public void Reset_Diem()
    {
    }
}
```

```

    {
        ThatBai_noti.SetActive(false);
    }
    public void Checked()
    {
        if (DapAn1 == true && DapAn2 == true && DapAn3 == true && DapAn4 == true)
        {
            Reset_btn.SetActive(false);
            Thanhcong_noti.SetActive(true);
            NhanBang_btn.interactable = true;
        }
        else
        {
            Reset_btn.SetActive(true);
            int Tong = Diem1 + Diem2 + Diem3 + Diem4;
            Diem.text = Tong.ToString() + "/4";
            ThatBai_noti.SetActive(true);
        }
    }
}

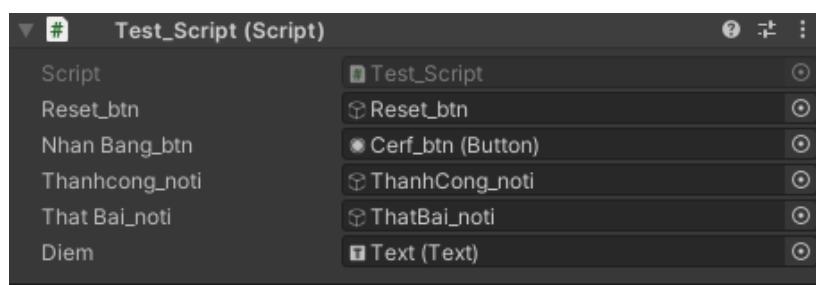
```

Khởi tạo biến Diem = 0 và gán chúng = 1 nếu Onclick() của Button được kích hoạt thông qua các hàm Answer() tương ứng. Hàm Checked() kiểm tra liệu biến “DapAn” của tất cả câu hỏi đã bằng “True” nếu thỏa mãn, thông báo “Hoàn thành” được hiển thị (*Hình 3.68*).



Hình 3.68. Thông báo hoàn thành (Bên trái) và chưa hoàn thành (Bên phải)

Nếu chưa đúng tất cả các câu hỏi, thông báo thất bại sẽ hiển thị cùng với nút “Làm lại” để đặt lại điểm và câu hỏi. Ở các Button câu trả lời, ta sẽ gán các C# Script vào sự kiện OnClick() và khai báo các đối tượng cần được cài đặt sự kiện (*Hình 3.69*).



Hình 3.69. Khai báo các đối tượng của sổ báo cáo để kích hoạt các sự kiện

CHƯƠNG 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI

4.1 KẾT LUẬN

Chương trình mô phỏng “Định lượng Chlorophyll bằng phương pháp đo quang phổ” được thiết kế và xây dựng đúng theo yêu cầu của giảng viên từ Viện Công nghệ Sinh học và Môi trường của Trường Đại học Nha Trang. Dựa vào môi trường Unity, tác giả đề tài đã xây dựng mô phỏng các bước thực hành của thí nghiệm. Tổng quan quy trình xây dựng dự án:

- Tìm hiểu về lĩnh vực Giáo dục số, khái niệm về bài giảng số và thí nghiệm ảo hỗ trợ của các bài giảng.
- Nghiên cứu và phân tích công cụ Game Engine phù hợp với đề tài.
- Tìm hiểu cách sử dụng các phần mềm hỗ trợ cho dự án như Adobe Illustrator, Photoshop, Blender, Camtasia, Vbee.vn,...
- Xây dựng kịch bản mô phỏng Unity dựa trên kịch bản thí nghiệm thực tế
- Xây dựng các vật thể 3D bằng phần mềm Blender
- Chuẩn bị các tệp âm thanh giọng nói AI bằng Vbee
- Thiết kế đồ họa, giao diện người dùng bằng Adobe Photoshop, Adobe Illustrator
- Cài đặt môi trường Unity cho dự án
- Xây dựng các phân cảnh dựa trên kịch bản thí nghiệm

Bảng 4.1. Bảng so sánh các tính năng của những dự án trong nước

Dự án	“Định lượng Chlorophyll bằng phương pháp đo quang phổ” – Nguyễn Tân Phát	“Bí quyết bảo quản thực phẩm” - BASF Kids’ Lab	“Chọn lọc tự nhiên” - PhET
WebGL	Có	Có	Có
Nền tảng 3D	Môi trường 3D nhưng Camera cố định	Chưa có	Chưa có
Tương tác đối tượng	Có	Có	Có

Hỗ trợ Tiếng Việt	Có	Có	Có
Data Player	Có	Có	Có
Câu hỏi trắc nghiệm	Có	Có	Chưa có
Model 3D Custom (Tùy chỉnh vật thể 3D)	Có	Chưa có	Chưa có
Hỗ trợ âm thanh	Có	Có	Chưa có
Môi trường mở	Chưa có	Chưa có	Có

Chương trình mô phỏng “*Định lượng Chlorophyll bằng phương pháp đo quang phổ*” đã được một số kết quả nhất định. Nhìn lại các sản phẩm mô phỏng trong nước, có thể thấy những điểm cải tiến của đề tài. Bên cạnh đó, có một số điểm cần cải thiện trong tương lai.

4.2 HƯỚNG PHÁT TRIỂN ĐỀ TÀI

Chương trình mô phỏng thí nghiệm “*Định lượng Chlorophyll bằng phương pháp đo quang phổ*” còn rất nhiều tính năng có thể thêm vào để hỗ trợ tốt hơn cho người dùng trong quá trình sử dụng chương trình mô phỏng:

- Thêm tính năng Gửi thông tin, kết quả về cho giảng viên
- Tạo môi trường mở, người dùng có thể tự do tương tác
- Xây dựng hệ thống chấm điểm đúng/sai theo từng bước thực hiện.
- Bảng điểm Highscore – liệt kê những người thực hiện bài tốt nhất.
- Thêm các hiện tượng sai khi làm sai bước.

Đề tài “*Định lượng Chlorophyll bằng phương pháp đo quang phổ*” là một trong những thí nghiệm tiêu biểu được thiết kế và xây dựng trong môi trường Game Engine. Xây dựng thí nghiệm ảo, chương trình mô phỏng là “mảnh đất màu mỡ” cho các lập trình viên do nhu cầu ngày càng lớn về các nền tảng giáo dục số ở các trường học. Vì

vậy, hi vọng trong tương lai gần, các dự án về bài giảng số, chương trình mô phỏng được triển khai nhiều hơn để giúp cho nền giáo dục nước nhà phát triển mạnh mẽ hơn nữa và bên cạnh đó, giúp tạo công việc cho những lập trình viên trẻ.

TÀI LIỆU THAM KHẢO

- [1] "*Common game development terms and definitions / Game design vocabulary / Unity*". Unity. Retrieved 2021-07-14.
- [2] "*The 10 Best Video Game Engines / 2018 Edition*". The Ultimate Resource for Video Game Design. 2017-03-11. Retrieved 2019-05-15.
- [3] "*Video Games Starting to Get Serious*". Gazette.net. 2007-08-31. Archived from the original on 2008-12-03. Retrieved 2011-01-17.
- [4] Schmidgen, Henning; Evans, Rand B. *The Virtual Laboratory: A New On-Line Resource for the History of Psychology*. *History of Psychology*, 6 (2), p. 208-213, 2003