國立臺東大學
資訊工程學系

112 學年度資訊專題成果報告書

阿虎的心願

指導教授：吳信德

學生：蘇韋嘉(10911119)

潘嘉茵(10911220)

中　　華　　民　　國　　1　1　2　　年　　1　2　　月

致謝辭

專題研究，儘管看似是一項相當艱鉅且個體化的任務，然而在這將近一年的深入探討與努力的過程中，我們深刻體會到，獲得了眾多導師和同學的悉心指導、熱心支持與簡貴建議。這段旅程中，我們有幸與各位尊師、尊長共同攜手，感受到他們的學術智慧與人生經驗的啟迪，使我們更深入地理解並突破研究的種種難關。

在此，我們要衷心感謝所有在我們專題研究過程中慷慨給予支持與建議的教職員工，感激各位同學間的合作共鳴，以及那些在我們陷入瓶頸時伸出援手的同儕夥伴。這份感激之情深刻地扎根於我們的心中，並成為我們成長歷程中難忘的一部分。

藉此機會，我們要特別表達對於我們學術道路上的引導者，給予無私指導與無盡耐心的導師們的深切謝意。感謝您們在我們每一步的前行中，不辭辛勞地給予教誨，讓我們在學術深淵中茁壯成長。

最後，我們想要向所有關心、支持並陪伴我們渡過這段研究之旅的人們致以最誠摯的謝意。在您們的協助與關懷下，我們能夠攜手克服種種挑戰，順利完成這項專題研究。期許未來，我們能將所學所得應用於實踐，回饋社會，並持續努力追求更高的學術成就。

摘要

蘭嶼，位於台灣東南方的島嶼，擁有獨特而豐富的生態特徵。其海域孕育著多樣性的海洋生物，包括珊瑚、海龜和各種熱帶魚類，形成獨特的海洋生態系。在陸地上，原生樹木和植被茂盛，而蘭嶼角鴞等特有鳥類成為島上珍貴的生態資源之一。

為了倡導保育這些瀕危物種，我們選擇採用 Unity 引擎來製作一款環境保育的遊戲，旨在幫助年輕的孩童更快速地認識這些瀕危物種。這款遊戲將透過生動的互動方式，呈現蘭嶼特有的生態環境，提高大眾對於自然保育以及蘭嶼特有種的關注度，並激發更多人參與保護這片寶貴的生態環境。

關鍵字：Unity、環境保育、蘭嶼、吻鰕虎

目次

第一章 緒論

第一節 研究背景及動機

一、 生態保育及研究

蘭嶼的生態系統得以有效保護，成為台灣島嶼生態多樣性研究的重要區域。這片獨特的土地不僅擁有美麗的自然景觀，更是生物多樣性研究和保育工作的寶貴資源，為我們深入了解自然環境提供了寶貴的機會。

二、 環境教育計畫

在一次與專題老師的談話中，我們偶然聊到關於保育生物的話題。隨著科技逐漸發展，人類的生活水平提升，但相對地也進行了大量開發項目，導致物種多樣性急速下降。為了防止出現比 6500 萬年前的第五次物種大滅絕更加嚴重的第六次物種大滅絕，我們認為環境保育的教育必須普及，尤其是在孩童時期。

三、 蘭嶼吻鰕虎的狀況

蘭嶼吻鰕虎魚是台東縣蘭嶼的特有種之一，在 2012 年天秤颱風重創蘭嶼之後，蘭嶼政府後續進行多項水利整治工程，尤其是水泥空心磚整治工程，嚴重影響蘭嶼吻鰕虎的兩側洄游的特殊生活方式，經過 2019 年生態專家周銘泰先生調查發現，蘭嶼吻鰕虎可能已經不足 200 隻，比起櫻花鉤吻鮭更加稀少，在台灣淡水魚紅皮書中被列入極度瀕危物種(CR)，可卻不在 1 級淡水魚保育名錄中，是典型不被大眾注意的瀕危物種。

為了喚起社會對蘭嶼吻鰕虎的關注，我們參考了蘭嶼吻鰕虎的繪本《阿虎的心願》並以此為基礎，製作了相應的環保教育遊戲。這款遊戲的目的在於透過互動的方式提高大眾對於蘭嶼吻鰕虎魚的認知，激發更多人參與相應的保育行動。

為實現這一目標，我們特地選擇 Unity 引擎。同時，我們訂立了另一項目標，即設計一款針對孩童的環保教育遊戲，希望透過遊戲引導孩子們更加關注、理解生態保育的重要性，培養環保意識，並激發他們參與保育行動的積極性。

透過以上三點努力，我們期望能夠為蘭嶼的生態保育和環保教育盡一份心力，喚起社會對於環境保護的重視，以促使這片富饒的

土地實現永續發展的目標。

第二節　研究目的

近年，全球物種多樣性的急遽下降已成為一項嚴重的環境議題。儘管如此，人們對環境保育的認知仍然面臨著種種挑戰。為因應這一現象，我們特別籌劃了一個以國小教育為主題的環境事件導向的動物保育教育遊戲。

本遊戲的主要研究目的在於透過互動式的方式，激發學童對物種多樣性和環境保育的興趣。我們原計畫是與學校合作，將這款簡單而有趣的小遊戲納入學校課程，特別是在電腦課程中透過老師的引導和協助，讓學生在課堂上共同參與這個遊戲，以實現教育目標。

此遊戲將提供有趣且具啟發性的遊戲場景，使學生能夠深入了解動物保育的重要性。同時，遊戲中的任務和挑戰將設計得兼具教育性質，貫穿知識學習的過程，培養學生對維護環境的積極性和主動參與的意識。

透過與學校的合作，我們期望推動環境保育意識，使更多學生透過遊戲中體會到保護生態環境的重要性。這不僅有利於培養下一代自然資源的保護意識，同時也為全球物種多樣性的保育盡一份心力。

第三節　研究限制

在研究進行中，我們面臨了一些限制，主要源於以蘭嶼吻鰕虎為基礎的遊戲，試圖介紹這種極度瀕危物種所面臨的生態威脅。原先計劃探索蘭嶼，蒐集部落傳統和進行實地考察。然而，由於蘭嶼地處偏遠，前往的時間和資金成本過大，我們不得不取消這一目標。

因此，我們決定重新調整研究方向，將焦點轉移到遊戲程式的優化上。我們將致力於修改遊戲程序，使其更加便利，同時縮小地圖大小，以提升遊戲的可訪問性。這樣的調整確保在不便利的環境下，孩子仍能輕鬆參與遊戲，進一步達到教育目標。

除此之外，我們還將重新利用遊戲故事內容，進行必要的修改，以確保故事更吸引並引起孩子的興趣。這包括調整故事情節，制定更貼近受眾目標文化和興趣的內容，提升遊戲整體的吸引力。

這一系列的調整旨在克服遠距離合作的挑戰，同時確保研究能夠在更實際和有效的全球環境中進行。我們期望這些改變能夠提升遊戲的教育效果，同時為未來類似研究提供有價值的經驗與啟示。

第四節　研究流程

一、 研究的出發點

1. 廣泛搜尋蘭嶼保護物種，資料收集讓我們深刻了解環境保護物種所面臨的嚴重挑戰。
2. 深入了解後發現，雖社會大眾對地球物種多樣性減少有認知，但積極參與環境保育活動的行為相對缺乏。

二、 思考與選擇

1. 如何引起更廣泛的社會關注，特別是針對極端瀕危物種，選擇以蘭嶼吻鰕虎為主題，一種典型且鮮為人知的瀕危物種。
2. 進行對蘭嶼吻鰕虎的深入研究，發現其在 2019 年已被警告為極度瀕危，卻未受到適當保育，可能已經滅絕。

三、 製作遊戲

1. 參考繪本的情節結構，逐一打造適合兒童的遊戲。

四、 結果與影響

1. 完成遊戲後，邀請適齡孩子進行測試，根據他們的回饋進行一系列修改，確保遊戲在引起孩童好奇心和同理心方面更為明顯。
2. 這過程豐富了研究方法，同時為遊戲的最終形成提供了實際的驗證。

第二章 文獻分析

第一節 遊戲式學習

一、 遊戲式學習的特性

1. 娛樂性：遊戲式學習提供愉悅與樂趣的體驗，提高內在動機，激發學習興趣。
2. 規則性：具有結構性，讓學習者更輕鬆地組織和理解遊戲內容，提升學習效果。
3. 目標性：以目標為核心，引導學習者進行任務，提供明確方向，增加參與動力。
4. 結果與回饋：提供即時的學習機會，制定調整策略，進一步提升學習成果。
5. 問題解決：遊戲情境中設定的問題激發創意思維，促使學習者在具體情境中思考解決辦法。
6. 圖像與情節性：透過豐富的圖畫和故事情節，使學習者獲得知識和情感層面的體驗。

二、 遊戲式學習的優勢

1. 激發內在動機：提高學習興趣，有助於記憶的保留。
2. 高層次思考：在遊戲情境中進行高層次的思考，不斷解決問題和做決策，促進深度參與。
3. 記憶與理解：教學內容巧妙佈局於遊戲中，使得學習者更容易記憶和理解。
4. 富有成效的學習形式：提供有趣且具啟發性的學習體驗，使學習成為一種成效卓越的過程。

第二節 環境保育

環境保育是當前全球共同關心的議題之一，隨著人類社會的不斷發展和自然環境的逐漸惡化，我們正面臨著前所未有的挑戰。為了更深入地理解和應對這些挑戰，我們特別進行了一系列文獻分析，從不同的視角探討環境保育的重要主題。

一、 生態系統管理和保護

深入研究各種生態系統的管理和保護策略，以維護生態平衡和生物多樣性。

二、 物種保育

探討瀕臨絕種或受威脅的動植物物種的保育方法，包括棲息地保護和保育計畫的執行。

三、 氣候變化和環境變遷

分析全球氣候變化的趨勢，詳細討論溫室氣體排放、海平面上升和極端天氣事件等相關研究。

四、 可持續發展

研究如何實現經濟、社會和環境的可持續發展，提出可行的政策和實踐建議。

五、 環境法規與政策

整理各國和國際層面上的環境法規和政策，評估其實施效果和潛在的改進空間。

六、 環境教育

探討如何透過環境教育提高公眾對環境問題的認識和參與，培養環保意識。

七、 環境科技和創新

分析新興科技和創新在環境監測、治理和保育方面的應用，評估其對環境可持續性的貢獻。

第三節　相關研究

一、 環境教育效果評估

評估了不同形式的環境教育對學生環保意識和行為的影響。通過課堂教學、戶外實地考察和互動式學習等方法，研究考察了環境教育的實際效果以及潛在的改進方向。

二、 戶外經驗對學習的影響

研究者調查了學生參與戶外環境教育活動後的學習成效。透過實地考察和自然環境中的互動，學生更容易理解生態系統、永續發展和環境變化等概念。

三、 跨學科環境教育

研究強調了跨學科方法在環境教育中的重要性。通過將科學、社會科學、藝術等不同學科整合到環境教學中，研究者試圖培養學生全面的環保意識。

四、 環境教育的技術應用

分析了科技在環境教育中的應用，包括虛擬實境、網路互動和行動應用。這些技術工具被視為提高學生參與和理解的有力手段。

第三章 專題設計及實作

第一節 專題架構

透過 Unity 作為開發平台，以 C#作為開發語言，參考最近環境與動物保育的相關文獻後，深刻了解環境與動物保育教育的重要性。在這基礎上，我計劃以蘭嶼吻鰕虎為故事主角開發一款遊戲，詳細呈現蘭嶼吻鰕虎的生活地點、生活方式，並介紹導致牠們瀕臨絕種的主要原因。

遊戲內容將包括詳盡的介紹，以文字方式呈現蘭嶼吻鰕虎的生態方式、瀕危原因，以及遊戲的前置故事。這樣一來，使用者能夠迅速融入蘭嶼吻鰕虎的視角，體驗族群為了生存而努力的情境。

遊戲全程都將有配樂，而一開始使用者會以蘭嶼吻鰕虎的視角開始冒險，試圖為族群尋找生存的希望。

然而，隨著故事發展，使用者會逐漸發現沒有其他動物能夠幫助牠們，最終族群滅亡。直到此時，使用者將發現傳說中的光澤蝸牛，這象徵著目前蘭嶼吻鰕虎所處的絕望情境。唯有人類停止破壞，牠們才有可能生存下去。

完成遊戲後，我將蒐集使用者的回饋，比較他們遊戲前後對蘭嶼吻鰕虎的認知程度，以評估遊戲是否成功引起使用者的興趣。同時，透過回饋，我將尋找遊戲中可能存在的問題，以便改善遊戲品質。

第二節 遊戲玩法

在遊戲的操作方面，我們設計了基本移動控制，使用 WASD 鍵來實現前進、後退、左移和右移的功能，同時透過 SPACE 鍵實現跳躍操作。此外，我們提供了左 SHIFT 鍵，玩家可透過它來加速移動，增添遊戲中的動態感。

遊戲中還有兩項技能，分別為「丟擲樹酯」和「回復水泡」。這兩項技能的選擇採用了鍵盤上的 1 和 2 鍵，玩家可輕鬆切換。選擇完技能後，只需按下左鍵即可使用，使玩家更加靈活地應對遊戲中的挑戰。

在與遊戲中的角色互動方面，我們提供了方便快捷的對話操作。玩家可以通過按下左鍵或 SPACE 鍵，以更迅速地進行對話，加深遊戲體驗。

這樣的操作設計不僅考慮了遊戲的實用性，同時也追求了流暢和直觀的操作體驗，以確保玩家在遊戲中能夠輕鬆自如地進行各種動作和互動。

第三節 專題內容

進入遊戲後，我們首先按序介紹蘭嶼吻鰕虎的棲息地、特徵、生活方式以及導致其瀕危的原因，最後揭開遊戲的前置故事，詳見圖 1 和圖 2。

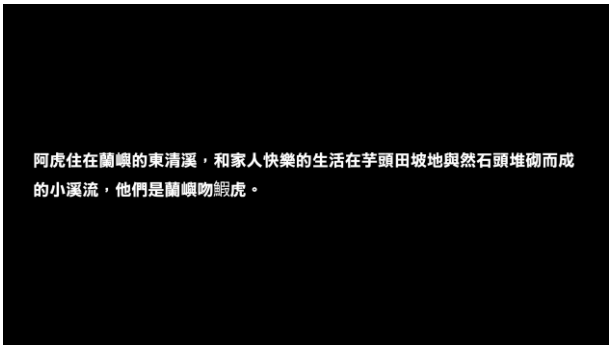

圖 1

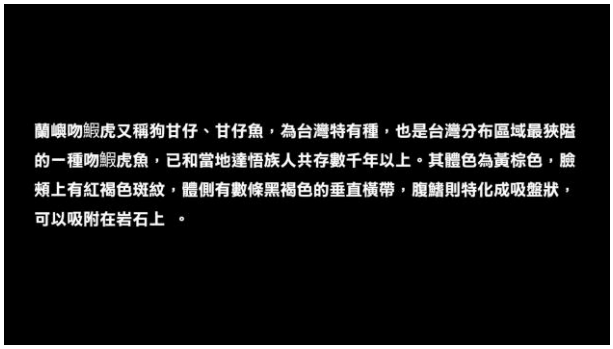

圖 2

正式進入遊戲後，玩家需要尋找畫面左邊的鱸鰻長老，按下 F 鍵進行對話。鱸鰻長老將告知玩家第一隻鳥的所在地，同時提供樹枝與水泡各 30 個作為道具。道具的使用方式是點擊鍵盤上的 1、2 鍵進行選擇，選定後點擊左鍵即可使用，詳見圖 3 和圖 4。

圖 3

圖 4

在遊戲過程中，玩家會目睹人類對森林的破壞。當接近人類時，將顯示文字提示，指示玩家使用樹枝來驅趕人類，阻止進一步的環境破壞。水泡的使用時機則為左上角藍色條狀過低時，使用水泡進行補充，詳見圖 5 和圖 6。

圖 5

圖 6

玩家的目標是找到三隻惡魔鳥，每隻都會告知玩家他們無法幫助蘭嶼吻鰕虎，同時提供下一隻惡魔鳥的位址。直到最後一隻惡魔鳥會介紹光澤蝸牛，告知玩家它能夠提供幫助。此時，文字通知玩家再過兩分鐘，蘭嶼吻鰕虎族群將會滅亡，需要迅速找到光澤蝸牛，詳見圖 7 至圖 10。

圖 7

圖 8

圖 9

圖 10

實際上，玩家找不到光澤蝸牛，因為它實際上象徵著天堂的使者。兩分鐘後，綠蠵龜會自動出現在玩家面前，引導玩家尋找光澤蝸牛，這象徵著蘭嶼吻鰕虎的滅亡。接著，文字描述了拯救蘭嶼吻鰕虎的方式，強調只有人類的停止破壞才能真正保護這一物種，詳見圖 11 和圖 12。

圖 11

圖 12

第四章 研究成果

第一節 成果

一、 國小環境保育教育的現狀

透過專題研究，我們深入了解了台灣國小教育中環境保育內容的現狀。雖然環保相關課程存在，但深度不足，無法深入觸及學童內心深處。尤其在國中階段，由於學生處於叛逆期，對於環境保育課程的受眾相對較少。

二、 專題遊戲的印象深刻性

通過實際遊戲測試，我們的專題遊戲在 8 位國小孩童中取得了顯著的成功。在陪同遊玩的情況下，這些孩童在一個月後仍能清晰記得蘭嶼吻鰕虎的相關重要內容，這顯示出專題遊戲在提升知識傳遞和印象深刻性方面的效果。

三、 互動式學習體驗

基於實際訪談結果，我們建議環境保育的教育內容應更早在國小時期實施。這樣不僅可以提高學童對環保議題的關注，還能在他們形成價值觀念的階段培養環保意識。專題遊戲的協助使得這一目標更為實現，提供一個具體、有趣且深具印象的學習平台。

四、 推動國小環境保育教育

我們的研究成果顯示，透過遊戲式學習可以有效地推動國小環境保育教育。建議進一步與學校合作，將專題遊戲納入正式教學計畫，提供教師培訓，並創建相應的教育資源，以擴大這種教育方法的應用範圍。

第二節 進一步研究之建議

透過我們對蘭嶼生態系的研究，我們志在不僅推廣環保意識，更透過向更廣泛的大眾、尤其是教育領域的傳播，將這份珍貴的知識深化植根於社會意識中。在持續進行更深入的研究時，我們將著眼於擴大研究範圍，納入更多元的生態系統和物種，這與我們一直以來的研究理念相契合。

將深入的生態研究轉化為適合不同年齡層的遊戲，我們希望能夠在教育領域發揮更大的影響力。我們計畫開發針對不同年齡層的

遊戲，以滿足不同學習階段的需求。這些遊戲將不僅僅侷限在吻鰕虎這一物種，而是以整個蘭嶼生態系統為基礎，將各種生物納入故事情節中。透過角色扮演和互動式學習，我們期望能夠啟發學童對於生態平衡和生物多樣性的熱情。

遊戲中的角色和故事情節將根據生態環境的差異進行調整，使學童能夠更全面地認識和保護各種生物。這種結合深度研究和教育遊戲的方法，不僅能夠使學童在遊戲中樂在其中，更能夠促使他們在現實中更主動參與生態保育的實踐。透過這樣的結合，我們期望能夠培養出更多對環境保護和生態平衡有深刻認識的未來世代。

第三節　未來展望

一、 遊戲教育的普及

我們計畫開發針對不同年齡層的生態教育遊戲，以豐富、有趣的方式呈現蘭嶼的生態多樣性。未來，我們期望這些遊戲能夠被廣泛應用於學校和其他教育機構，推動生態教育的普及。

二、 多元化遊戲內容

我們將努力擴展遊戲的內容，不僅關注吻鰕虎等特定物種，還包括整個蘭嶼生態系統中的各種生物。透過多元的遊戲情節和角色，使玩家能夠全面認識並保護蘭嶼豐富的生態資源。

三、 互動式學習體驗

我們致力於創造更互動式的學習體驗，讓玩家透過遊戲不僅能夠獲取知識，更能夠參與保護行動。這種互動性有望激發玩家對生態保育的主動參與，培養他們的環保意識。

四、 技術創新與遊戲平台

隨著科技的進步，我們將不斷探索新的技術應用，如擴增實境（AR）或虛擬實境（VR），以提升遊戲的沉浸感和教育效果。同時，我們將利用不同的遊戲平台，包括手機應用、電腦遊戲等，讓更多人能夠輕鬆參與生態教育。

第五章 總結

第一節 結論

透過我們的專題研究，深刻體悟到現今教育體制在教學方法上仍偏向傳統的「紙上談兵」，缺乏真實世界的實地考察與實際操作的體驗。然而，實踐教育並非易事，牽涉到龐大的經費開支、學生安全的保障以及時間成本的考慮，使得這種方式難以在效益上取得平衡。

在科技進步的當下，我們認為遊戲式學習成為突破這一困境的最佳途徑之一。遊戲式學習能夠透過引人入勝的遊戲場景，引起學習者極高的興趣，結合教育目標設計豐富有趣的遊戲內容，使學習者更加專注於所學內容。

第二節 心得

透過遊戲式學習，我們不僅提供了知識，更深刻地意識到進行環境與動物保育不能僅止於紙本與電視螢幕之間。真正解決保育問題需要讓人們在心中擁有深刻的體驗，而非僅僅是一時的憐憫。我們深信，遊戲式學習是解決環境與動物保育問題的重要而關鍵的方法之一。透過這種互動性、引人入勝的學習方式，我們有望激發更多人對於環保和生態保育的認識，從而共同努力保護地球上珍貴的生態系統。

最後，這次的專題研究讓我深信遊戲式學習是未來教育的重要方向。透過引人入勝的遊戲體驗，我們能夠更有效地傳達知識、激發學習興趣，並在受測者中培養對於環保議題的關懷。我期待這樣的教學方式能夠被更廣泛地應用，為未來的教育帶來更多可能性。

參考文獻

中研院數位文化中心(2013)。**台灣魚類資料庫。**2023 年 1 月 27 日，取自 https://fishdb.sinica.edu.tw/chi/species.php?id=381841。

王維聰、王建喬(2011)。**數位遊戲式學習系統。**科學發展，467，46-51。

王維聰、王建喬、林櫻蓮(2023)。**以資訊回饋與科技接受模式觀點探討影響大學學生採用遊戲式學習系統之關鍵因素:以電子化資料庫正規化教學為例。**資訊管理學報，30:4，347-376。

林德、曾萬年、葉明峰(2014)。**台灣產吻鰕虎屬魚類的形態與分布模式。**台灣生物多樣性研究，16:2，095-116。

胡通哲、陳淑媛、陳彥旭、陳兆鈿(2022)。**河川水域與濱溪棲地品質分析－以水璉溪為例。**土木水利:中國土木水利工程學會會刊，49:2，021-026。

莊明德、周文杰、吳宗儒(2012)。**最適河川棲地綜合適合度指數之選取研究-以筏子溪為例。**水保技術，7:4，228-236。

楊孟軒(2018 年 01 月 09 日)。**【台灣紅皮書】特殊生活史遭水泥工程阻斷「兩側洄游」的蘭嶼吻鰕虎。**社團法人台灣環境資訊協會、行政院農業委員會林務局。2023 年 2 月 15 日，取自 https://e-info.org.tw/node/209397。

楊善媛、高錦瑗、黃國禎(2014)。**由失敗經驗探討數位遊戲式學習系統設計的考量因素。**數位學習科技期刊，6:4，37-51。

蕭如真(2021)。**傳統智慧與知識在環境保育中的角色。**農訓雜誌，373，054-056。

顧玉蓉(2020)。**向暖爸致敬。短吻紅斑吻鰕虎。**中華防災學刊，12:1，111-118。

附錄

Menu.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
public class Menu : MonoBehaviour
{
    public Button startButton;
    public Button exitButton;

    private void Start()
    {
        Cursor.lockState = CursorLockMode.None;
        Cursor.visible = true;
        startButton.onClick.AddListener(OnStartButtonClick);
        exitButton.onClick.AddListener(OnExitButtonClick);
    }

    public void OnStartButtonClick()
    {
        SceneManager.LoadScene(1);
    }

    public void OnExitButtonClick()
    {
        Application.Quit();
    }
}
```

CameraController.cs

```
using UnityEngine;
using System.Collections;

public class CameraController : MonoBehaviour
{

    // When to update the camera?
    [System.Serializable]
    public enum UpdateMode
    {
        Update,
        FixedUpdate,
        LateUpdate,
        FixedLateUpdate
    }

    public Transform target; // The target Transform to follow
    public Transform rotationSpace; // If assigned, will use this Transform's rotation as the rotation space instead of the world space. Useful with spherical planets.
    public UpdateMode updateMode = UpdateMode.LateUpdate; // When to update the camera?
    public bool lockCursor = true; // If true, the mouse will be locked to screen center and hidden

    [Header("Position")]
    public bool smoothFollow; // If > 0, camera will smoothly interpolate towards the target
    public Vector3 offset = new Vector3(0, 1.5f, 0.5f); // The offset from target relative to camera rotation
    public float followSpeed = 10f; // Smooth follow speed

    [Header("Rotation")]
    public float rotationSensitivity = 3.5f; // The sensitivity of rotation
    public float yMinLimit = -20; // Min vertical angle
```

```
    public float yMaxLimit = 80; // Max vertical angle
    public bool rotateAlways = true; // Always rotate to mouse?
    public bool rotateOnLeftButton; // Rotate to mouse when left button is pressed?
    public bool rotateOnRightButton; // Rotate to mouse when right button is
pressed?
    public bool rotateOnMiddleButton; // Rotate to mouse when middle button is
pressed?

    [Header("Distance")]
    public float distance = 10.0f; // The current distance to target
    public float minDistance = 4; // The minimum distance to target
    public float maxDistance = 10; // The maximum distance to target
    public float zoomSpeed = 10f; // The speed of interpolating the distance
    public float zoomSensitivity = 1f; // The sensitivity of mouse zoom

    [Header("Blocking")]
    public LayerMask blockingLayers;
    public float blockingRadius = 1f;
    public float blockingSmoothTime = 0.1f;
    public float blockingOriginOffset;
    [Range(0f, 1f)] public float blockedOffset = 0.5f;

    public float x { get; private set; } // The current x rotation of the camera
    public float y { get; private set; } // The current y rotation of the camera
    public float distanceTarget { get; private set; } // Get/set distance

    private Vector3 targetDistance, position;
    private Quaternion rotation = Quaternion.identity;
    private Vector3 smoothPosition;
    private Camera cam;
    private bool fixedFrame;
    private float fixedDeltaTime;
    private Quaternion r = Quaternion.identity;
    private Vector3 lastUp;
    private float blockedDistance = 10f, blockedDistanceV;
```

```
    public static CameraController Instance;

    public void SetAngles(Quaternion rotation)
    {
        Vector3 euler = rotation.eulerAngles;
        this.x = euler.y;
        this.y = euler.x;
    }

    public void SetAngles(float yaw, float pitch)
    {
        this.x = yaw;
        this.y = pitch;
    }

    // Initiate, set the params to the current transformation of the camera relative to the target
    protected virtual void Awake()
    {
        Vector3 angles = transform.eulerAngles;
        x = angles.y;
        y = angles.x;

        distanceTarget = distance;
        smoothPosition = transform.position;

        cam = GetComponent<Camera>();

        lastUp = rotationSpace != null ? rotationSpace.up : Vector3.up;

        Instance = this;
    }

    protected virtual void Update()
```

```
    {
        if (updateMode == UpdateMode.Update) UpdateTransform();
    }

    protected virtual void FixedUpdate()
    {
        fixedFrame = true;
        fixedDeltaTime += Time.deltaTime;
        if (updateMode == UpdateMode.FixedUpdate) UpdateTransform();
    }

    protected virtual void LateUpdate()
    {
        UpdateInput();

        if (updateMode == UpdateMode.LateUpdate) UpdateTransform();

        if (updateMode == UpdateMode.FixedLateUpdate && fixedFrame)
        {
            UpdateTransform(fixedDeltaTime);
            fixedDeltaTime = 0f;
            fixedFrame = false;
        }
    }

    // Read the user input
    public void UpdateInput()
    {
        if (!cam.enabled) return;

        // Cursors
        Cursor.lockState = lockCursor ? CursorLockMode.Locked :
CursorLockMode.None;
        Cursor.visible = lockCursor ? false : true;
```

```
        // Should we rotate the camera?
        bool rotate = rotateAlways || (rotateOnLeftButton &&
Input.GetMouseButton(0)) || (rotateOnRightButton && Input.GetMouseButton(1)) ||
(rotateOnMiddleButton && Input.GetMouseButton(2));

        // delta rotation
        if (rotate)
        {
            x += Input.GetAxis("Mouse X") * rotationSensitivity;
            y = ClampAngle(y - Input.GetAxis("Mouse Y") * rotationSensitivity,
yMinLimit, yMaxLimit);
        }

        // Distance
        distanceTarget = Mathf.Clamp(distanceTarget + zoomAdd, minDistance,
maxDistance);
    }

    // Update the camera transform
    public void UpdateTransform()
    {
        UpdateTransform(Time.deltaTime);
    }

    public void UpdateTransform(float deltaTime)
    {
        if (!cam.enabled) return;

        // Rotation
        rotation = Quaternion.AngleAxis(x, Vector3.up) * Quaternion.AngleAxis(y,
Vector3.right);

        if (rotationSpace != null)
        {
            r = Quaternion.FromToRotation(lastUp, rotationSpace.up) * r;
```

```
            rotation = r * rotation;

            lastUp = rotationSpace.up;

        }

        if (target != null)
        {
            // Distance
            distance += (distanceTarget - distance) * zoomSpeed * deltaTime;

            // Smooth follow
            if (!smoothFollow) smoothPosition = target.position;
            else smoothPosition = Vector3.Lerp(smoothPosition, target.position,
deltaTime * followSpeed);

            // Position
            Vector3 t = smoothPosition + rotation * offset;
            Vector3 f = rotation * -Vector3.forward;

            if (blockingLayers != -1)
            {
                RaycastHit hit;
                if (Physics.SphereCast(t - f * blockingOriginOffset,
blockingRadius, f, out hit, blockingOriginOffset + distanceTarget - blockingRadius,
blockingLayers))
                {
                    blockedDistance = Mathf.SmoothDamp(blockedDistance,
hit.distance + blockingRadius * (1f - blockedOffset) - blockingOriginOffset, ref
blockedDistanceV, blockingSmoothTime);
                }
                else blockedDistance = distanceTarget;

                distance = Mathf.Min(distance, blockedDistance);
            }
```

```
            position = t + f * distance;

            // Translating the camera
            transform.position = position;
        }

        transform.rotation = rotation;
    }

    // Zoom input
    private float zoomAdd
    {
        get
        {
            float scrollAxis = Input.GetAxis("Mouse ScrollWheel");
            if (scrollAxis > 0) return -zoomSensitivity;
            if (scrollAxis < 0) return zoomSensitivity;
            return 0;
        }
    }

    // Clamping Euler angles
    private float ClampAngle(float angle, float min, float max)
    {
        if (angle < -360) angle += 360;
        if (angle > 360) angle -= 360;
        return Mathf.Clamp(angle, min, max);
    }
    public void SwitchRotateCamera(bool b)
     {
        rotateAlways = b;
     }
}
```

PlayerController.cs

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using System.Collections;

public class PlayerController : MonoBehaviour
{
    public static PlayerController Instance;

    [SerializeField] public float walkSpeed = 3f;
    [SerializeField] public float jumpHeight;
    [SerializeField] public float rotationSpeed = 0.1f;

    private bool isGround;
    [SerializeField] private float groundCheckDistance;
    [SerializeField] private LayerMask groundMask;

    [SerializeField] private float gravity;
    private CharacterController controller;

    private float speedsmoothVelocity = 0f;
    private float speedSmoothTime = 0.1f;

    private float currnetspeed = 0f;

    private Vector2 moveinput;
    private Vector3 velocity;

    private Animator anim;
    private Camera cam;

    private bool autoRun;
```

```
private bool hasBubble;
public ParticleSystem[] bubbles;
public GameObject addBubbleParticle;

public Transform shootPos;

public bool canMove = true;

public Slider hpSlider;
private float currentHp;
public float maxHp;

public Slider bubbleSlider;
private float currentBubble;
public float maxBubble;

void Start()
{
	Instance = this;
	cam = Camera.main;
	anim = GetComponent<Animator>();
	controller = GetComponent<CharacterController>();
	controller.detectCollisions = false;

	currentHp = maxHp;
	hpSlider.maxValue = currentHp;
	hpSlider.value = currentHp;

	currentBubble = maxBubble;
	bubbleSlider.maxValue = currentBubble;
	bubbleSlider.value = currentBubble;
}

void Update()
{
```

```
        CheckAddBubble();

        if (!canMove)
            return;

        if (Input.GetKeyDown(KeyCode.V))
        {
            SwitchBubble(!hasBubble);
        }
        if (Input.GetKeyDown(KeyCode.Equals))
        {
            autoRun = !autoRun;
        }
        float h = Input.GetAxisRaw("Horizontal");
        float v = Input.GetAxisRaw("Vertical");
        if(h !=0 ||v != 0)
        {
            autoRun = false;
        }
        moveinput = autoRun ? new Vector2(0, 1) : new Vector2(h, v);

        isGround = Physics.CheckSphere(transform.position,
groundCheckDistance, groundMask);
        velocity.y += !isGround ? gravity * Time.deltaTime : 0;

        Vector3 forward = cam.transform.forward;
        Vector3 right = cam.transform.right;
        forward.y = 0;
        right.y = 0;
        forward.Normalize();
        right.Normalize();
        Vector3 desiredMoveDirection = (forward * moveinput.y + right *
moveinput.x).normalized;

        if (desiredMoveDirection != Vector3.zero)
```

```
        {
            transform.rotation = Quaternion.Slerp(transform.rotation,
Quaternion.LookRotation(desiredMoveDirection), rotationSpeed);
        }
        float targetspeed = walkSpeed * moveinput.magnitude;
        bool runInput = Input.GetKey(KeyCode.LeftShift);
        targetspeed = runInput ? targetspeed * 2 : targetspeed;
        anim.speed = runInput ? 1 : 0.5f;

        currnetspeed = Mathf.SmoothDamp(currnetspeed, targetspeed, ref
speedsmoothVelocity, speedSmoothTime);

        controller.Move(desiredMoveDirection * currnetspeed * Time.deltaTime);

        controller.Move(velocity * Time.deltaTime);
        anim.SetBool("Move", moveinput.sqrMagnitude != 0);

        if (Input.GetKeyDown(KeyCode.Space) && isGround)
        {
            velocity.y = jumpHeight;
        }
    }

    public void SwitchBubble(bool b)
    {
        if (b)
        {
            hasBubble = true;
            for (int i = 0; i < bubbles.Length; i++)
            {
                var pt = bubbles[i].main;
                pt.loop = true;
                bubbles[i].Play();
            }
        }
```

```
        else
        {
            hasBubble = false;
            for (int i = 0; i < bubbles.Length; i++)
            {
                var pt = bubbles[i].main;
                pt.loop = false;
            }
        }
}

public void EnableMoveing(bool b)
{
        canMove = b;
        if (!canMove)
        {
            anim.SetBool("Move", false);
        }
}

public void OnGetHit(float value)
{
        currentHp -= value;
        hpSlider.value = currentHp;
        if (currentHp <= 0)
        {
            Debug.LogError("End");
            StartCoroutine(StartToMenu());
        }
}

IEnumerator StartToMenu()
{
        yield return new WaitForSeconds(5f);
        SceneManager.LoadScene("Menu");
```

```
    }

    public void UpdateBubble(float value)
    {
        if (!canMove)
            return;

        currentBubble += value;
        bubbleSlider.value = currentBubble;
        currentBubble = Mathf.Clamp(currentBubble, 0, maxBubble);
        if (currentBubble <= 0)
        {
            currentHp -= Time.deltaTime;
            hpSlider.value = currentHp;
            if (currentHp <= 0)
            {
                Manager.Instance.gameOverUI.SetActive(true);

                Debug.LogError("End");
                StartCoroutine(StartToMenu());
                /*if (Input.anyKeyDown) {
                    SceneManager.LoadScene(1);
                }*/

            }
        }
    }

    private void CheckAddBubble()
    {
        bool inWater = transform.position.y <= -1.4f;
        addBubbleParticle.SetActive(inWater);
        UpdateBubble(inWater ? Time.deltaTime * 1 : Time.deltaTime * -1);
    }
}
```

HumanMale.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class HumanMale : MonoBehaviour
{
    private Animator anim;

    private bool isOver;

    private Vector3 runTarget;

    private bool startRun;

    public float runSpeed = 10;

    public AudioSource hit;

    void Start()
    {
        anim = GetComponent<Animator>();
    }

    private void Update()
    {
        if (startRun)
        {
            Vector3 dir = runTarget = transform.position;
            Quaternion lookDir = Quaternion.LookRotation(dir);
            transform.position += transform.forward * runSpeed *
Time.deltaTime;
            transform.rotation = Quaternion.Slerp(transform.rotation, lookDir,
Time.deltaTime * 6);
        }
```

```
    }

    public void GetHit(float heightY)
    {
        if (isOver)
            return;
        if (heightY >= 1f)
        {
            anim.SetBool("Run", true);
            runTarget = Manager.Instance.runPos[Random.Range(0,
Manager.Instance.runPos.Length)].position;
            hit.Play();
            startRun = true;
            isOver = true;
        }
        else
        {
            if (!anim.GetCurrentAnimatorStateInfo(0).IsName("Look"))
            {
                anim.SetTrigger("Look");
            }
        }
    }
}
```

Manager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Manager : MonoBehaviour
{
    public static Manager Instance;

    public Transform[] runPos;

    private int findBirdCount;

    public GameObject[] finalBirdBlock;

    public GameObject gameOverUI;

    void Start()
    {
        Instance = this;
    }
    private void Update()
    {
        if (Input.GetKeyDown(KeyCode.Tab))
        {
            ItemSystem.Instance.SwitchBag();
        }
    }

    public void FindBird(int index)
    {
        findBirdCount++;
        if(findBirdCount == 3)
        {
            finalBirdBlock[index - 1].SetActive(true);
```

```
            }
        }
    }
```

Branch.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Branch : MonoBehaviour
{
	private Rigidbody rb;
	public float power;
	public float torque;

	private bool isHit;

	public void Init(Vector3 forward)
	{
		rb = GetComponent<Rigidbody>();
		rb.AddForce(forward * power);
		rb.AddRelativeTorque(new(500,0,500));
		Destroy(gameObject, 5);
	}

	private void OnCollisionEnter(Collision collision)
	{
		if (isHit)
			return;

		if (collision.collider.CompareTag("Human"))
		{
			HumanMale humanMale =
collision.collider.GetComponent<HumanMale>();
			humanMale.GetHit(transform.position.y);
			isHit = true;
		}
	}
}
```

FastItemGrid.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class FastItemGrid : MonoBehaviour
{
	public Image selectImage;
	public Text countText;

	public Outline outline;

	public void UpdateValue(int value)
	{
		countText.text = $"{value}";
	}
	public void Select(bool b)
	{
		outline.enabled = b;;
	}
}
```

ItemContainer.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[CreateAssetMenu(menuName = "Config/ItemData")]
public class ItemContainer : ScriptableObject
{
    [System.Serializable]
    public struct Item
    {
        public string itemName;
        public int itemIndex;
        public Sprite itemPhoto;
    }
    public Item[] items;
}
```

ItemGird.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class ItemGird : MonoBehaviour
{
    private int index;
    private int itemIndex;
    private int count;
    public Text countText;

    public Image itemPhoto;

    public void Init(int _index)
    {
        index = _index;
        itemPhoto.enabled = false;
        countText.gameObject.SetActive(false);
    }

    public void JoinNewItem(int index , int count , Sprite sprite)
    {
        itemIndex = index;
        itemPhoto.sprite = sprite;
        itemPhoto.enabled = true;
        countText.gameObject.SetActive(true);
        AddItem(index , count);
    }

    public void AddItem(int index , int _count)
    {
        count += _count;
        countText.text = $"{count}";
```

```
            ItemSystem.Instance.UpdateFastItemGird(index, count);
        }

        public int GetItem()
        {
            return itemIndex;
        }

        public int GetCount()
        {
            return count;
        }
        public bool IsNull()
        {
            return itemIndex == 0;
        }
}
```

ItemSystem.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ItemSystem : MonoBehaviour
{
    public static ItemSystem Instance;

    private bool isOpen;

    public GameObject bagPanel;
    public Transform girdRoot;

    public ItemGird itemGird;

    private List<ItemGird> itemGirds = new();

    public const int itemGirdCount = 16;

    public ItemContainer itemContainer;

    public FastItemGrid[] fastItemGrids;

    private int currentSelect = 1;

    [Space]
    public Branch branch;
    [Space(10)]
    public GameObject bubbleFx;

    public AudioSource uiSfx;

    public void Start()
    {
```

```
        Instance = this;
        for (int i = 0; i < itemGirdCount; i++)
        {
            ItemGird gird = Instantiate(itemGird, girdRoot);
            gird.Init(i);
            itemGirds.Add(gird);
        }
        UpdateCurrentSelect(1);
    }
    private void Update()
    {
        if (!PlayerController.Instance.canMove)
            return;

        if (Input.GetKeyDown(KeyCode.Alpha1))
        {
            UpdateCurrentSelect(1);
        }
        if (Input.GetKeyDown(KeyCode.Alpha2))
        {
            UpdateCurrentSelect(2);
        }
        if (Input.GetMouseButtonUp(0))
        {
            UseItem();
        }
    }

    private void UseItem()
    {
        if (!PlayerController.Instance.canMove)
            return;

        if (GetItemCount(currentSelect) <= 0)
            return;
```

```
            GetItem(currentSelect, -1);
            if (currentSelect == 1)
            {
                Transform point = PlayerController.Instance.shootPos;
                Branch _branch = Instantiate(branch, point.position,
Quaternion.identity);
                _branch.Init(point.forward);
            }


            if(currentSelect == 2)
            {
                GameObject fx = Instantiate(bubbleFx,
PlayerController.Instance.transform.position, bubbleFx.transform.rotation);
                Destroy(fx, 3);
                PlayerController.Instance.UpdateBubble(50);
            }
        }

        public void SwitchBag()
        {
            isOpen = !isOpen;
            bagPanel.SetActive(isOpen);
            CameraController.Instance.rotateAlways = !isOpen;
            PlayerController.Instance.EnableMoveing(!isOpen);

            Cursor.lockState = !isOpen ? CursorLockMode.Locked :
CursorLockMode.None;
            Cursor.visible = !isOpen ? false : true;
        }

        public void GetItem(int index , int count)
        {
            for (int i = 0; i < itemContainer.items.Length; i++)
```

```
        {
            if(itemContainer.items[i].itemIndex == index)
            {
                for (int q = 0; q < itemGirds.Count; q++)
                {
                    if(itemGirds[q].GetItem() == index)
                    {
                        itemGirds[q].AddItem(index ,count);

                        return;
                    }
                    else
                    {
                        if (itemGirds[q].IsNull())
                        {
                            itemGirds[q].JoinNewItem(index, count,
itemContainer.items[i].itemPhoto);
                            return;
                        }
                    }
                }
            }
        }

    }

    public int GetItemCount(int index)
    {
        return itemGirds[index - 1].GetCount();

    }

    public void UpdateFastItemGird(int index , int value)
    {
        fastItemGrids[index - 1].UpdateValue(value);
```

```
    }

    public void GetTree(int count)
    {
        GetItem(1, count);
    }
    public void GetBubble(int count)
    {
        GetItem(2, count);
    }

    private void UpdateCurrentSelect(int index)
    {
        uiSfx.Play();
        currentSelect = index;
        for (int i = 0; i < fastItemGrids.Length; i++)
        {
            fastItemGrids[i].Select(false);
        }
        fastItemGrids[index - 1].Select(true);
    }

}
```

PosRandomer.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PosRandomer : MonoBehaviour
{
	public Transform[] targetPos;
	void Start()
	{
		transform.position = targetPos[Random.Range(0,
targetPos.Length)].position;
	}
}
```