MYP Personal Project - Video Game Development

Objective A: Planning

I. Learning Goal and Personal Interest

I love logic and numbers flipping around my head explains my great interests in mathematics. When I first encountered video games, I found fun in calculations of skills, techniques, and strategies. At a certain point, I began wanting to manipulate the numbers behind the screen instead of what is shown to me, meaning to become a programmer myself. This summer, I learned the basics of building settings and adding codes in Unity, a popular game engine, also the one used to develop the very first games I encountered. With the extra time apart from classes this semester, I can further research on programming and turn my creativity into actual video games.

In this project, I plan to learn the balance between work on front and back ends and how to communicate with the game developing engine with scripts, from manipulating visual displays, object-oriented programming (OOP), to cross file programming.

II. Product Goal

Given my learning goals, the most practical way to achieve and the most useful way to implement the learnings would be actually developing a video game and learning from the process. Therefore, I plan to write code scripts and design artwork for a video card game. In a document separate from the game, I will explain the logic of the codes. The requirements for the game and document are in the following section.

III. Success Criteria

I divided up the success criteria by “front end”, “back end”, or both, depending on whether each element of the product is visible to users. This classification is common for computer-related products, from video games to websites, for convenience in division or work and labor. Although I am responsible for the quality of both front and back ends, both artistic and technical areas, setting success criteria for both fields demonstrates my understanding and capability for any position in a game developing team.

Index	Category	Statement	Requirement & Explanation
1	Front End	The game rule and user interface can be easily interpreted by players in the expected way.	The final version receives positive feedback about interpretation of game rules and user interface, or functions of buttons and panels. Game rules should be included in game settings, i.e. not in an external file. Game rules and user interface easy to understand are basics of a successfully playable game. Therefore, users should be able to understand and play the game with information given in the game, without extra teachings.
2		The art design is clear, and the meanings can be easily interpreted by players in the expected way.	The final version receives positive feedback about interpretation of images. Distortion from low resolution during display is minimized. This is the only success criterion for art elements, as the project focuses on the coding in game development. The quality of visual elements not only affects the engagement of players but also represents the developer’s respect towards the product.
3		The display includes smooth 2D and 3D animations.	Include movements of both 2D and 3D gameobjects in Unity. Embedded videos are additional and not necessary. Animation increases the engagement of players by adding sensory feedback to the process of “user input → screen output”. The lack of smoothness and variety of animation decreases such impact on user engagement.
4	Back End	The student fully understands and can explain	Create a mind map explaining the code structures used in the development of the product, separated by functions of the algorithms, including OOP and cross file programming. Pseudo codes or C#

		object-oriented programming (OOP) and cross file programming in C#, Unity, and other algorithms.	codes are allowed to give examples, but terms cannot be accurately understood without coding experience require explanations and are not recommended. When programming, copying scripts from manual or open sources is common and justified, but copying without understanding does not qualify a person as a game developer. Therefore, this test on full understanding is necessary to link my learning and product goals.
5	Front & Back End	The game, from visual displays to input/output process, functions in the expected way.	The final release does not contain unintended bugs, or all identified errors are eliminated. Malignant bugs limit the enjoyment from video games and worsen the user experience. Therefore, the product should be tested through multiple beta versions and collections of feedback, and the programmer should modify the codes in response.

IV. Plan and Associated Success Criteria (S.C.)

I proposed an action plan with specific date(s) and required resources listed for each task. I determined the order based on time and sources required for each stage of development. For example, research precedes coding, and art design precedes application of art through scripts.

Date	Task	Detail	S.C. Index	Tools & Sources
2022/10/15-2022/10/16	2D movement research	Research built in methods in Unity for 2D translation, rotation, and acceleration of the motions.	3,4	Unity Manual Microsoft Learn - C# Youtube Tutorial StackOverflow
	2D movement experiment	Build a 2D model in Unity to check functionalities of methods found.	3,4	Unity (2022.3.30f1) Visual Studio
2022/10/22-2022/10/23	Research 3D movement control in Unity	Research built in methods in Unity for 3D translation, rotation, and acceleration of the motions and detecting collisions.	3,4	Unity Manual Microsoft Learn - C# Youtube Tutorial StackOverflow
	Create experimental model of 3D movement in Unity	Build a 3D model in Unity to check functionalities of methods found.	3,4	Unity (2022.3.30f1) Visual Studio
2022/10/29	Card deck form finalization	Determine the display form of card decks as gameobjects and card categories.	1	Unity (2022.3.30f1)
	Game rule finalization	Finalize the content and descriptions of game rules to insert in the game.	1	Google Doc
2022/11/5-2022/11/6	Randomizing algorithm research	Research algorithms to generate random series, lists or numbers.	4	Youtube Tutorial StackOverflow
	Shuffle algorithm	Use randomizing algorithms found to design algorithm shuffling cards.	4	Visual Studio

	Object-Oriented Programming (OOP) research	Research C# OOP and related built in tools in Unity.	4	Microsoft Learn - C# Youtube Tutorial StackOverflow
	Cross file programming research	Research cross file programming in Unity.	4	Microsoft Learn - C# Youtube
	Algorithm design (non script form)	With text descriptions or mind maps to present structures of codes about to use in actual scripts.	4	Google Doc
2022/11/20	Visual Art Design	Complete the design of the cards' back and front. Unify formats of descriptions and images.	2	Paint Tool SAI
2022/11/26-2022/11/27	Research art conversion in Unity	Research built in methods in Unity to apply and control textures of gameobjects with completed art designs.	2	Unity Manual Youtube Tutorial
	Experiment art conversion in Unity	Use gameobjects from current progress or a separate project and apply the completed art designs.	2	Unity (2022.3.30f1) Visual Studio
2022/12/3	Rule-unrelated scripts	Create scripts for element displays and attach to materials.	3,4	Unity (2022.3.30f1) Visual Studio
2022/12/24	Rule script and application	Create scripts for calculations and attach to scenes.	4	Unity (2022.3.30f1) Visual Studio
2022/12/31	Apply Art Design	Apply all art designs in the completed scripts.	2	Unity (2022.3.30f1) Visual Studio
2023/1/1	Feedback collection	Share the execution file of the beta version through email and social media. Attach a form to collect user feedback.	1,2,5	Gmail LINE Discord Google Form
2023/1/14	Error organization	List and categorize errors identified through playing or from recipients of the product file.	5	Google Form Google Spreadsheet Google Doc
2023/1/15	Debugging	Debug according to list of identified errors	5	Unity (2022.3.30f1) Visual Studio
	Modify game rule and art design	Modify descriptions of game rules and art designs according to feedback on interpretation.	1,2	Google Doc Paint Tool SAI
2023/1/20	Final release	Share the final product through email and social media.	1,2,5	Gmail LINE Discord

Objective B: Applying Skills

I. Achieving Learning Goal

ATL Skills: Research (Information & Media Literacy), Self-Management (Organizational)

My research for learning goals alternated with the creation of my product due to several properties of my learning goal, which consists of learning the balance between art designs and programming during a video game development, and the methods to write and run scripts through a game developing engine. Firstly, the balance between art and programming is a subjective conclusion based on efforts spent for different areas in the product. Next, communications with a game development engine have multiple ways and uncertainties to be tested and selected when making the product. The methods and errors to research turned out to exceed the action plan in folds when I began the actual game development. The following tables present the most fundamental parts of my research. During the long research, I not only applied the skills of finding and organizing information, but also developed information and media literacy and learned a new standard of selecting platforms and information under the context of programming.

Methods and Algorithms:

Goal: Make GameObjects move or rotate		
Source	Suggested Method	Adopted & Reason
Unity Manual[1][2][3] docs.unity3d.com/ScriptReference/Vector3.html docs.unity3d.com/ScriptReference/Quaternion.html docs.unity3d.com/ScriptReference/MonoBehaviour.InvokeRepeating.html	Change the position and rotation of GameObjects with Vector3 and Quaternion following the declaration instruction, and repeatedly change towards a direction with InvokeRepeating	Adopted as meeting the needs, and the declarations of Vector3 and Quaternion have sufficient efficiency to coordinate with other parts of script
My modification: Make parameters for Vector3 and Quaternion global variables to modify as InvokingRepeating cannot run functions with parameters		

Goal: Drag and drop GameObjects		
Source	Suggested Method	Adopted & Reason
YouTube[4] www.youtube.com/watch?v=yalbvB84kCg	Track the mouse position on screen and set GameObject to the mouse position	Not adopted because the method does not apply to 3D GameObjects, and making cards 2D limits the animations
YouTube[5] www.youtube.com/watch?v=uNCCS6DjebA	Track the mouse position with Raycast and set GameObject to the position	Adopted as meeting the needs and allowing other variables to be added for reference
My modification: Move the section updating per frame from script Grabber to script SlotAndCard		

Goal: Shuffle card decks (arrays)		
Source	Suggested Method	Adopted & Reason
GeeksforGeeks[6] www.geeksforgeeks.org/shuffle-a-deck-of-cards-3/	For every position in the array, swap it with another position determined by random.	Adopted as meeting the needs, and the logic and code structure are clear, simple, and readable
My modification: none		

Errors:

Error message: Object reference not set to an instance of an object

Occurrence upon:
1. Clicking on GameObject for decorations in the background
2. Checking if the name property of an EVO is blank
3. Checking if the EVO property of a SlotContent is null

Source	Explanation	Suggested Solution	Adopted & Reason
Unity Forum[7] forum.unity.com/threads/object-reference-not-set-to-an-instance-of-an-object-c.226430/	The object does not exist.	Make sure there is an object with the name	Not adopted because it is not the case and problem I met
StackOverflow[8] stackoverflow.com/questions/64589974/nullreferenceexception-object-reference-not-set-to-an-instance-of-an-object-un	The script is trying to refer to an object that is empty.	Make sure all variables have set values and are not null	Adopted as meeting the need and critically points out the mistake I made
quest.com[9] www.quest.com/community/migration-manager-for-ad/f/forum/31022/object-reference-not-set-to-an-instance-of-an-object	The Object referred to by the “object reference” does not exist or was deleted or cleaned up.	Check if the object is null with conditional statements before using	Not adopted because I need to use the object without error, not just to avoid the error

My explanation of the error:
1. The clicked GameObjects were duplicated from the card template with the draggable tag but without properties necessary to further run the drag-and-drop.
2. Instead of being blank, the name property of EVO does not exist as the EVO does not exist.
3. Instead of being null, the EVO property of SlotContent does not exist as the SlotContent does not exist.

My solution:
1. Remove the draggable tags from the GameObjects
2. Instead of checking the name property, check if the EVO is null; as EVOs always have name properties upon declaration, the circumstances of name being blank and EVO being null are equivalent
3. Initiate the SlotContents when runtime begins

Error message:
You are trying to create a MonoBehaviour using the ‘new’ keyword. This is not allowed.

Occurrence upon:
1. Creating a MonoBehaviour SlotAndCard inside class SloContent in order to refer to other classes and variables declared in SlotAndCard

Source	Explanation	Suggested Solution	Adopted & Reason
StackOverflow[10] stackoverflow.com/questions/40640553/why-in-unity-im-getting-the-warning-you-are-trying-to-create-a-monobehaviour-u	MonoBehaviours are Components to attach to GameObjects and cannot be created.	Create a GameObject and use AddComponent to attach the MonoBehaviours	Not adopted because generating a GameObject just to refer to classes and variables in MonoBehaviour is unnecessary

My explanation of the error:
Unlike built-in and self-defined classes, such as GameObject and SlotContent, MonoBehaviours, or scripts, cannot be accessed alone without being attached to a GameObject running the methods defined in the MonoBehaviour.

My solution:
Change the datatype of parameters in conditional statements from EVO (a self-defined class in SlotAndCard) to string, more specifically the name property of the EVO, so that accessing EVO variables declared in SlotAndCard is unnecessary

As shown above, different sources have various explanations and vagueness and do not guarantee to meet my demand. Therefore, filtering the sources is crucial for efficiency in research and development. In the case of programming, interactive forums where people ask questions for others to answer turned out to be more useful than officials or authorities, which is counterintuitive as opposing the principle of past scientific research. Unofficial discussions often include more specific cases and readable solutions, and the engine enables immediate tests of information, making coding research differ from regular research. As a result, platforms including StackOverflow, GeeksforGeeks, and Quora take up more proportion in my research than the Unity and Microsoft Manuals. However, video sources, YouTube in this case, were less helpful than the manuals. Although videos provide step-to-step guidance and screen view during the development, the information takes more time to obtain, understand, and test. Not until having watched the videos completely could I determine the ways to test and whether I could adopt the methods. Moreover, capturing certain sections of information from videos with progress bars is harder and less efficient than from scrollable web pages.

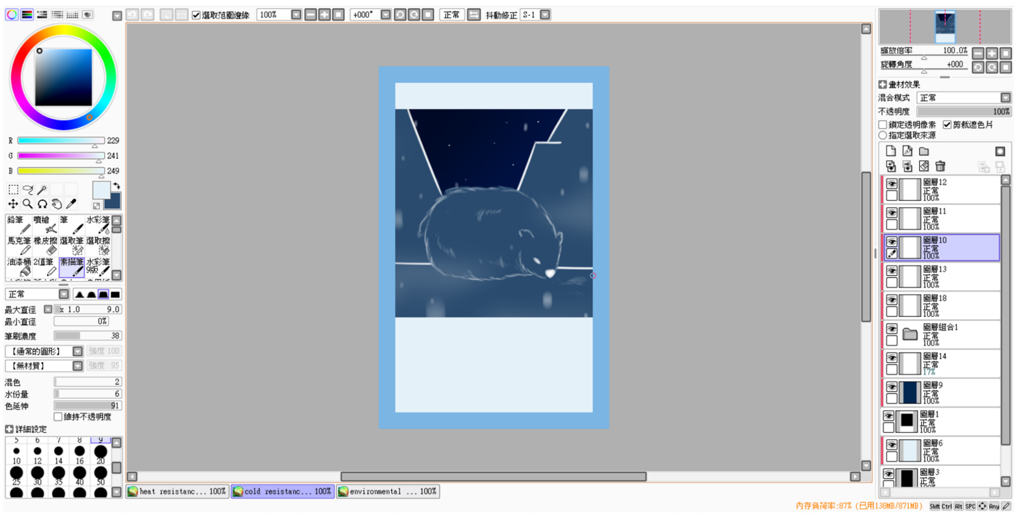
During and before the research, I reached a quick conclusion about the balance of art versus code: within standards below masterpiece, coding takes more time and effort than art designing. While scripts must be rigorously correct, visual art, without correct answers, allows more inaccuracy. Moreover, improving the image after completing the base leads to unproportionally much time spent on little effects. Therefore, I did not do extra research and spent much less time on the art work. I am not very surprised, but a little disappointed with the answer to the first half of my learning goal. I am quite satisfied with the second half, however, as the development went smoother and smoother towards the end.

II. Achieving Product Goal

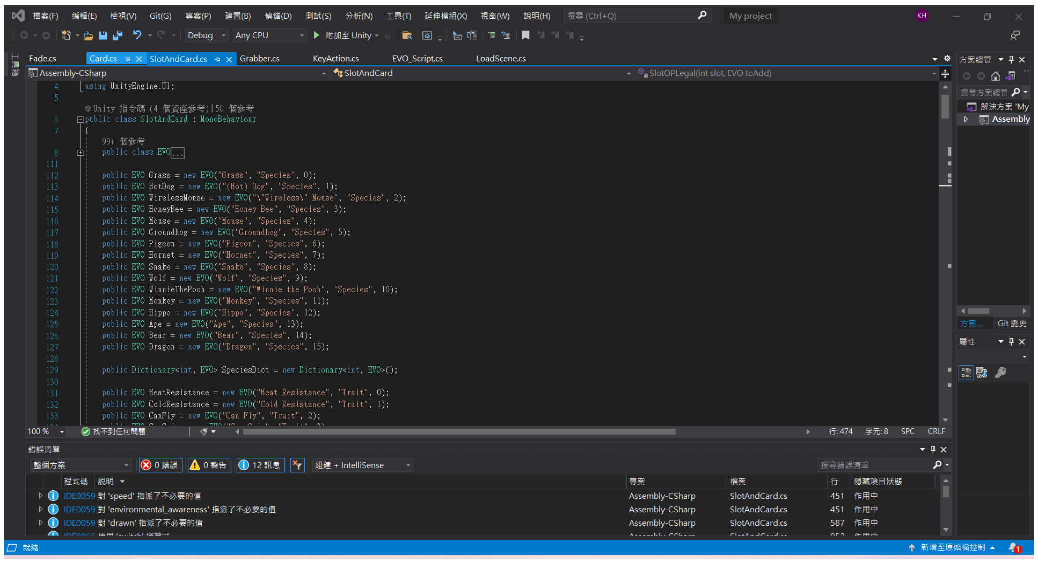
ATL Skills: Self-Management (Organizational), Thinking (Creative & Critical Thinking, Transfer)

My product goal also consists of two halves, the execution file of the video game and a document explaining the code implemented. The video game can further divide into art designs and scripts.

The art design consists of creative thinking more than other ATL skills, especially for the design of trait cards. While species and effects are easy to find references and draw, traits, as invisible or abstract concepts are hard to visualize. In the end, I determined, as species are the core of the game, I would demonstrate the traits with associated animals. This did not apply to traits *Immunity* and *Trait Extension*, however, because immunity is not a trait specific species would have, and trait extension is not a real trait. Below is the view in Paint Tool SAI, the drawing tool I used for the card designs. The card in the image is the trait *Cold Resistance* symbolized with a polar bear sleeping in snowy ice mountains.



For the game back end, while the information from forums were readable and testable, the testing from source to source was still time consuming, challenging, and frustrating, especially when finding multiple solutions from the research that were not compatible with each other. As a result, I developed the skill of establishing more effective methods of organization. I divided up the scripts, and in each script I separated the functions and variables by purposes. For example, in the script SlotAndCard, I have the self-defined class “EVO” and functions to create and initialize the “EVOs” in the top section, variables related to hand types in the next section, etc., and functions to actually run during runtime in the final section. The following image evidences my script organization. The different scripts are shown in the tabs on top, and the main screen is the first section, “EVO”, of the script SlotAndCard.


檔案(F) 編輯(E) 檢視(V) Git(G) 專案(P) 建置(B) 偵錯(D) 測試(S) 分析(N) 工具(T) 延伸模組(X) 視窗(W) 說明(H)
Debug
Any CPU
附加至 Unity
Fade.cs
Card.cs
SlotAndCard.cs
Grabber.cs
KeyAction.cs
EVO_Script.cs
LoadScene.cs
Assembly-CSharp
SlotAndCard
SlotOPLegal(int slot, EVO toAdd)

```
using UnityEngine.UI;

Unity 指令碼 (4 個資產參考)|50 個參考
public class SlotAndCard : MonoBehaviour
{
    99+ 個參考
    public class EVO...

    public EVO Grass = new EVO("Grass", "Species", 0);
    public EVO HotDog = new EVO("(Hot) Dog", "Species", 1);
    public EVO WirelessMouse = new EVO("\"Wireless\" Mouse", "Species", 2);
    public EVO HoneyBee = new EVO("Honey Bee", "Species", 3);
    public EVO Mouse = new EVO("Mouse", "Species", 4);
    public EVO Groundhog = new EVO("Groundhog", "Species", 5);
    public EVO Pigeon = new EVO("Pigeon", "Species", 6);
    public EVO Hornet = new EVO("Hornet", "Species", 7);
    public EVO Snake = new EVO("Snake", "Species", 8);
    public EVO Wolf = new EVO("Wolf", "Species", 9);
    public EVO WinnieThePooh = new EVO("Winnie the Pooh", "Species", 10);
    public EVO Monkey = new EVO("Monkey", "Species", 11);
    public EVO Hippo = new EVO("Hippo", "Species", 12);
    public EVO Ape = new EVO("Ape", "Species", 13);
    public EVO Bear = new EVO("Bear", "Species", 14);
    public EVO Dragon = new EVO("Dragon", "Species", 15);

    public Dictionary<int, EVO> SpeciesDict = new Dictionary<int, EVO>();

    public EVO HeatResistance = new EVO("Heat Resistance", "Trait", 0);
    public EVO ColdResistance = new EVO("Cold Resistance", "Trait", 1);
    public EVO CanFly = new EVO("Can Fly", "Trait", 2);
```

100 % 找不到任何問題 行: 474 字元: 8 SPC CRLF

錯誤清單

整個方案 0 錯誤 0 警告 12 訊息 組建 + IntelliSense

程式碼	說明	專案	檔案	行	隱藏項目狀態
IDE0059	對 'speed' 指派了不必要的值	Assembly-CSharp	SlotAndCard.cs	451	作用中
IDE0059	對 'environmental_awareness' 指派了不必要的值	Assembly-CSharp	SlotAndCard.cs	451	作用中
IDE0059	對 'drawn' 指派了不必要的值	Assembly-CSharp	SlotAndCard.cs	587	作用中

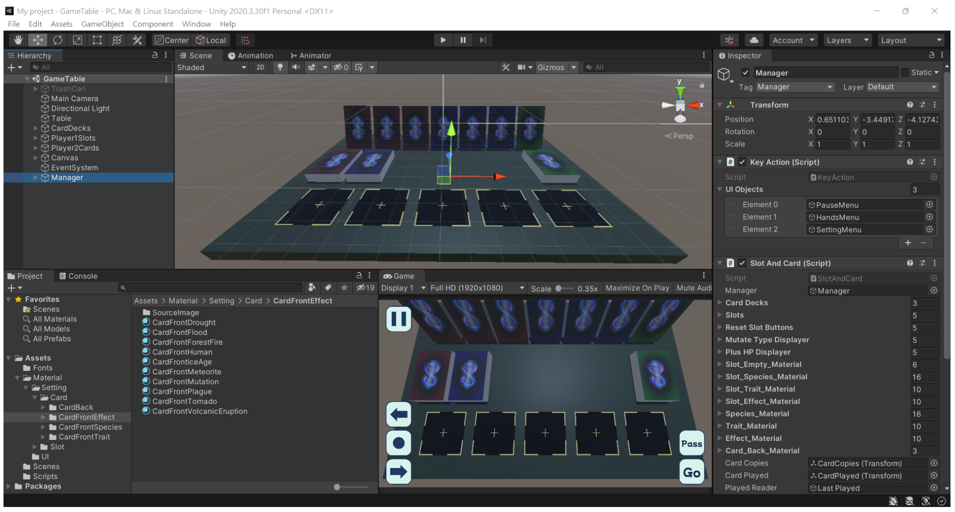
就緒 新增至原始檔控制

This separation is more useful than alphabetical order when testing, combing, and adding my own codes into methods found in the research. However, codes designed by myself, which takes up the majority, as well as my edits in the inspector screen often generate errors, presented in the research section above. The most serious and frequent error was “Object reference not set to an instance of an object”. The error stopped any further operations, so I could not check things after the error-generating line before I solved the problem. Although as shown in the research section, I found a feasible solution, I applied the solution in the wrong way that made my scripts run without errors but lose functionality.

Moreover, I tried to pass and use EVO type variables through functions, such as creating an EVO *a* that equals the existing EVO *Grass*, which I thought would create a clone of *Grass* with the name *a*. However, this actually makes the names *a* and *Grass* refer to the same EVO, meaning modification to *a* would also change the properties of *Grass*. I took a long time to understand this phenomenon and still could not find a clear solution.

For both the first problem, scripts without errors but also functionality, and the second, changing properties of objects I do not want to change, I used the Debug.Log function in all possible if/else paths to identify where the script begins to run not as expected. Even after identifying the location of errors, I spent much time understanding the logic of my codes that caused the malfunction, but the long thinking led to the moment I demonstrated my critical thinking and problem solving skills the most. In the two sections with errors, despite the different purpose and the different types of operations, I saw the similarity in one of their possible solutions: I need a new, complete object, an object with complete parameters, and new so distinct from all the other objects. Therefore, I came up with initialization along declaration as the solution to both problems, creating a new object so it would not be null, creating a new object so its name would not refer to objects I did not want to modify, and my previous learning allowed me to construct the codes to create and initialize self-defined objects with the efficiency and convenience of OOP. This is the critical point in my coding, as I had not thought of initialization not in the beginning of running codes before. After I solved the two major problems, I transferred the concepts and structure to other sections and began using this method more frequently in functions such as clearing slots and restarting the game.

In the end, I was able to expand my scripts quickly and build my scenes. The following picture shows the scene I built in the engine. The list on the left shows objects created and displayed (some are hidden, such as the pause menu); the inspector on the right shows components of the object, including text, renderer, and the scripts I typed and attached.



Hierarchy
- GameTable
  - TrashCan
  - Main Camera
  - Directional Light
  - Table
  - CardDecks
  - Player1Slots
  - Player2Cards
  - Canvas
  - EventSystem
  - Manager

Project
- Favorites
  - Scenes
  - All Materials
  - All Models
  - All Prefabs
- Assets
  - Fonts
  - Material
    - Setting
      - Card
        - CardBack
        - CardFrontEffect
        - CardFrontSpecies
        - CardFrontTrait
      - Slot
    - UI
  - Scenes
  - Scripts
- Packages

Assets > Material > Setting > Card > CardFrontEffect
- SourceImage
- CardFrontDrought
- CardFrontFlood
- CardFrontForestFire
- CardFrontHuman
- CardFrontIceAge
- CardFrontMeteorite
- CardFrontMutation
- CardFrontPlague
- CardFrontTornado
- CardFrontVolcanicEruption

Inspector
Manager — Static
Tag Manager — Layer Default

Transform

Position	X	0.651103	Y	-3.44917	Z	-4.12743
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

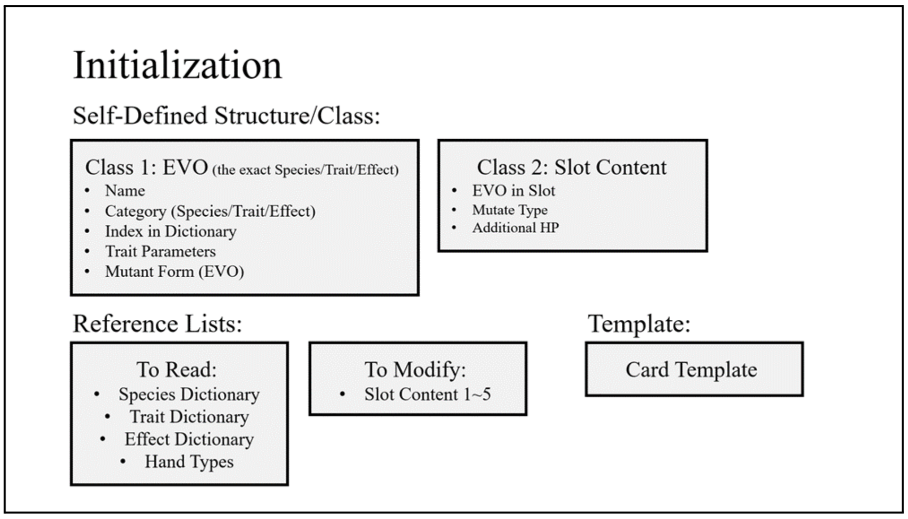
Key Action (Script)

Script	KeyAction
UI Objects	3
Element 0	PauseMenu
Element 1	HandsMenu
Element 2	SettingMenu

Slot And Card (Script)

Script	SlotAndCard
Manager	Manager
Card Decks	3
Slots	5
Reset Slot Buttons	5
Mutate Type Displayer	5
Plus HP Displayer	5
Slot_Empty_Material	6
Slot_Species_Material	16
Slot_Trait_Material	10
Slot_Effect_Material	10
Species_Material	16
Trait_Material	10
Effect_Material	10
Card_Back_Material	3
Card Copies	CardCopies (Transform)
Card Played	CardPlayed (Transform)
Played Reader	Last Played

The last part of the product, a separate document explaining the logics used in the development of the game, came straightforward and less time-consuming, as the document was a transfer and translation of what I had already typed in the scripts, and I had the organized information from the research to help with the accuracy of explanations. The following picture shows a sample page from the document, listing out objects I created in the codes. For example, the first block below the title *Self-Defined Structure/Class* represents the EVO class mentioned before, which was not default and was created by myself. As shown in bullet points in the block, an EVO has a name, belongs to a category, has an index in the dictionary, has specific traits and corresponding levels, and has a mutant form (if the EVO is a species).



Initialization

Self-Defined Structure/Class:

Class 1: EVO (the exact Species/Trait/Effect)
- Name
- Category (Species/Trait/Effect)
- Index in Dictionary
- Trait Parameters
- Mutant Form (EVO)

Class 2: Slot Content
- EVO in Slot
- Mutate Type
- Additional HP

Reference Lists:

To Read:
- Species Dictionary
- Trait Dictionary
- Effect Dictionary
- Hand Types

To Modify:
- Slot Content 1~5

Template:

Card Template

Objective C: Reflecting

I. The Impact of Project on Learning

By the completion of this project, I have reached the answers to my learning goals. Comparing the amount of time, I came to a definite conclusion about the balance between arts and codes: coding overwhelmingly required more time, efforts, and resources than art design in a video game, and as I have completed the product successfully, as evaluated in the below section, I can confidently claim I have learned the way to communicate with the game developing engine with codes with the use of object oriented programming and cross-file references.

As intended for the learning goal, I applied and strengthened the skills of and became Thinkers, Knowledgeable, and Balanced by creating a video game from codes to art designs. Covering both artistic and scientific aspects made me a balanced learner, while the constructions and tests with the codes trained me as a knowledgeable thinker. Challenges appeared as expected before the project, and my quick understanding and thinking carried me through the obstacles. However, beyond the expectation, I also developed the skill of being Open-Minded, or even Risk-Takers. Rather than being stuck in past experiences, I adapted, or experimented with new standards and methods when researching and programming.

As mentioned in the applying skills section, I learned a new standard and procedure of selecting information in a research. As the information I looked for was testable and was used as steps of my creation instead of quotes to insert in writings, the credibility of the author and platform providing the information became less significant than I would consider if conducting research for written works of science or social studies, meaning I could include information from more sources with lower depths on my list. In terms of planned research preceding actual creation, this decision increases the amount of time for organization and reduces the efficiency of the testing stage. However, also as mentioned in the applying skills section, the research for programming was dynamic due to uncertainties during the creation of the product, and no beforehand research could be guaranteed used or not used in the product. For the same reason, I had encountered insufficient research in past creative assignments before, where I ended up abandoning plans and searching for guides online arbitrarily with poor organization. Nonetheless, I learned, from the personal project, to modify instead of simply ignoring the schedule.

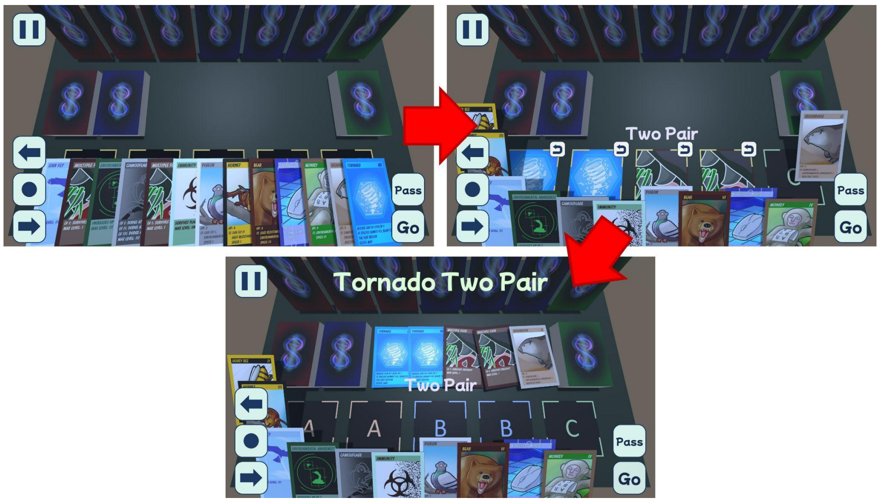
I kept the research divided up into stages, such as 2D movement testing, 3D movement testing, and rendering, but I combined the time block for the research and product creation. Moreover, I allowed changes in the order of the blocks. For example, I could research on and test for 2D movement before working on rendering, or if I found the process of rendering included the foundation of 2D movements, I could work on scripts for rendering first. These two changes gave more flexibility and reduced chances of redundant preparations, and as I still followed a schedule, despite its seeming disorder, I made steady progress and finished my product creation before the deadline. This experience impacted my attitude towards research plans, which I was obliged to make in past social studies assignments but did not know how to handle without the confidence to strictly follow the plan. Now having learned the way to construct an action plan for dynamic research, I will be able to better organize my timelines, from static research for an essay, dynamic preparation for an art design, to study for an upcoming exam, reducing the constant time pressure I had been facing and improve the quality of my works and learning. From a broader perspective, the project gave me more confidence in trying new approaches. Thus in the future, I will be willing to, therefore surly and efficiently, even more creatively, invest time in my greater, riskier goals.

II. Product Evaluation

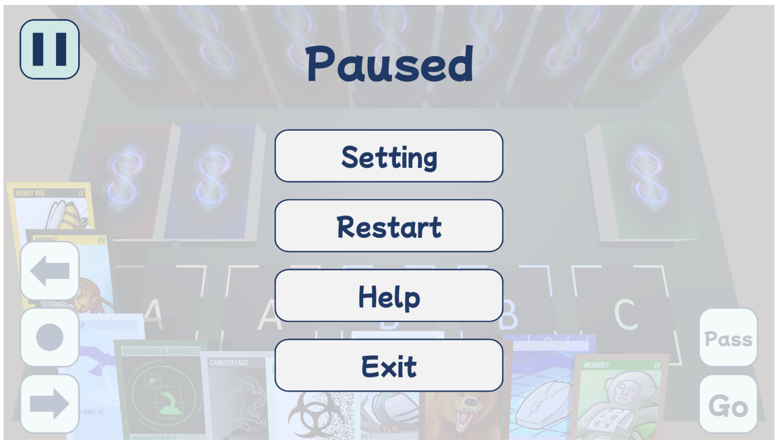
My product includes the files of the video game and a document explaining the codes without using coding languages. The following explains how my product functions in points, while further details and game rules could be found inside the game:

- Upon running the execution file and loading the scene, the game automatically draws 13 cards as the opening. Clicking on one card makes the card follow the cursor, and clicking again drops the card. If dropping the card on a slot where the card is legal to be placed, the card would be inserted into the slot, and a button with a return arrow appears on the top right corner of the slot, clicking which returns all cards inserted in the slot. Clicking on one of the 3 card decks, *Species* (blue), *Trait* (red), and *Effect* (green), draws 1 card in the corresponding category.
- The top left button pauses the game.
- The bottom left buttons changes the hand type, including *four of a kind*, *full house*, *pair*, etc., like in poker; the left arrow changes forward, or to the next bigger hand, and the right arrow changes the opposite way; the circle opens a menu of hands for the player to select by name.
- The bottom right button reading *Go* plays the cards in the slots if the cards in slots match the hand type and the hand is legal to play based on the previous hand on the table. The bottom right button reading *Pass* clears the table, and the previous hand on the table becomes empty. Whether a hand can be played after another hand is determined based on the game rules, available from the game.

The following screenshots show views at different steps in the game, starting with the automatic draw of 13 cards, then placing cards into slots according to the selected hand type, in this case *Two Pair*, and then playing the hand, in this case *Tornado Two Pair*.



The following image shows the paused view in the game. *Setting* directs to a page to change the maximum number of cards the player can hold (the default is 16); *Restart* resets and shuffles the card deck and clears the table, but does not make the automatic draw of 13 cards; *Help* directs to a Google Document with the game rules; *Exit* turns off the game and closes the window.



The pause button in top left, the play button in bottom right, the circle representing menus, and the card decks giving cards when clicked were all designs I found in common in the video games I played, and I inherited the features to make the interface more intuitive to players.

The document explaining the codes, as shown an excerpt in the applying skills section, covers firstly the main elements I created in the scripts, including the self-defined class EVO and the EVO dictionaries, secondly the main checks during the game, including whether a card can be placed in a slot and whether a hand can be played, thirdly the tree diagram determining whether a hand can be played, and finally the life cycle of a card object. I purposely selected the four aspects to include, each with a specific role in the explanation. The self-defined class was the core of object oriented programming, which was one of the focuses in my learning goal, also the most special section that I wanted to share the most to the audience; the two main checks brought the topic back to conventional coding with simple conditional statements, reemphasize coding as the core of my project with readable, friendly displays; the tree diagram introduced the core game rules, expanding from the introduced conditional statements, reminding the audience how deep conventional coding could reach; lastly, the life cycle of a card object demonstrates a unique feature of coding with a game developing engine, and closed the explanation with a relatively casual ending.

I conducted a survey within school to collect feedback after sharing the game file and the explaining document. The following table is the evidence of my survey, an excerpt of the spreadsheet recording my survey results for the evaluation with success criteria. Each scoring has a range from 1 to 8. *Previous Knowledge of the Game* was for identifying biases in the results; *Game UI Understanding* and *Game Rule Understanding* were used for success criterion 1; *Art Design Identification* and *Art Design Connection* were used for success criterion 2; *Animation Smoothness* was used for success criterion 3; *Code Explanation Clarity* was used for success criterion 4; success criterion 5 was evaluated independently from the survey.

Response Source	Previous Knowledge of the Game	Game UI Understanding	Game Rule Understanding	Animation Smoothness	Art Design Identification	Art Design Connection	Code Explanation Clarity
S11352	Yes	8	8	8	8	7	7
S11262	No	8	8	8	8	8	8
S11419	No	8	8	8	8	8	8
S11229	No	8	7	8	8	7	7
S11293	No	8	7	8	8	8	8
S11349	Yes	8	8	7	8	8	6

The following table lists evaluations based on the survey above and the success criteria mentioned in the planning section of the report, referencing the statements. The detailed requirements and reasons behind can be referred to from the success criteria part in the planning section.

Index	Statement	Rate (1~8)	Reason	Actions I did/should do that lead to the success
1	The game rule and user interface can be easily interpreted by players in the expected way.	8	Leaving out ones with previous knowledge of the game, the majority and rounded average of respondents' scores for game UI and game rule understanding are 8 out of 8, and the in-person verification afterwards proved their understanding correct.	I introduced the game rules in detail, covering every possible scenario before questions came up; I used clear, common symbols for the buttons, and thus, the respondents could use the interface intuitively.
2	The art design is clear, and the meanings can be easily interpreted by players in the expected way.	8	The majority and rounded average of respondents' scores for art design identification and connection are 8 out of 8. Art design identification means if the respondent can tell what is in each picture, and connection means if the respondent can relate the picture to the name of the card, especially for the *Traits*.	I used animals known with specific traits as symbols to convey the abstract traits of organisms. I referred to online sources and several photographs I had taken to draw clear pictures with accurate structures, such as *Human* (*Effect*) the bird in *Can Fly* (*Trait*).
3	The display includes smooth 2D and 3D animations.	8	Drawing and moving cards both involved 2D and 3D animations; the majority and the rounded average of the respondents' scores for animation smoothness are 8 out of 8.	The research on the InvokeRepeating function in Unity and choice of simple translations and rotations prevented lags and resulted in smooth animations.
4	The student fully understands and can explain object-oriented programming (OOP) and cross file programming in C#, Unity, and other algorithms.	7	The majority of the respondents' scores for code explanation clarity is 8 out of 8, and the rounded average is 7. Success in teaching or explaining concepts to a person without the base knowledge is the best evidence of understanding.	I fully understood the logic in the game rules and made sufficient corresponding research. I also connected the C# coding with past experiences on Sololearn introduced in computer programming class last year.

5	The game, from visual displays to input/output process, functions in the expected way.	7	There are no distorted or broken textures in the game; the UI operations functioned as introduced above; all features of the game rules were tested and proved possible to occur, in the correct way.	I did thorough research for rendering and wrote complex yet complete logic loops in the scripts. I minimized unnecessary variables, functions, and operations to avoid lagging from limited resources.