Ved Dhiman

12SDD Assessment Task 1 - 2022.pdf - Google Drive

Defining the Problem.

There is currently a lack of fun offline games that specifically address friends who wish to play a game centered on mathematical identities. Therefore, I propose the creation of a card game (or several), which uses mathematics and existing games as a basis for its (their) rules and functionality. This game will be targeted towards groups of mathematicians or those who enjoy mathematics and find games like Uno to be bland and lacking mathematics. An example of such a revision could be Complex Uno, where players must match a mathematical condition on the most recently played card, or Pi Blackjack, where cards must multiply to below a value.

Design specifications

To fulfil this functionality, the game must first obtain input from users. Pygame is able to do this, so this is simple to do. The game must also track the hands of all players, and their scores. This can be done using classes, storing each card in an array attribute, and each score as a float or integer attribute. The game must also display each player's hand on their turn, requiring the ability to display cards, and a visual to display as representative of a card. This can again be done using classes, having a common Card class for each card, storing their value, suit, visual asset path, and other important attributes. The user must have some indication of the cards on the "table". The cards on the table can be stored as an array of Card objects, displaying the topmost card as per the card's icon. Additionally, if further exploration elements are implemented, There must be a means to represent a player's sprite, and to use the controls to move the player, and interact with other sprites, which can be done by plotting each sprite as per coordinates, hiding the sprite if it is outside the range of visible coordinates.
Using these concepts as a basis, I will be making a card game compilation, possibly with open-world elements, that will be mathematically modified, with the aim to fill a niche in the mathematical digital card game market.

Scripting Language Being Used, and Why.

My game will be coded in the high-level language Python. This is because of several reasons, such as my experience with the language, the compatibility and low computer resource cost, and the multitude of PyPi and builtin libraries available. I have been coding with Python over the past year, and therefore am familiar with its simple syntax and its useful shorthand features, such as walrus expressions, and list comprehension, and I am also familiar with the structure of classes and modules.

Features of Python

The structure of Python deviates greatly in aspects from other languages, such as in the use of whitespace rather than curly braces, and in the syntax of boolean comparison, where Python uses use the words “and”, “or”, and “not” for boolean, making it far more readable than other languages. Furthermore, Python is largely modular, and is executed line-by-line, rather than compiled, which means that Python scripts will be structured differently to those of other languages which compile scripts. This is also a severe disadvantage of Python, since Python projects will be distributed as direct source files, which can easily be datamined, and/or altered and redistributed. Python is also not as efficient at multithreading as other languages, such as C# and C++ are. Python is an Object-Oriented Programming Language, but also allows structured and functional programming, which makes it easy to develop in. Python is able to run on many operating systems, such as Windows and MacOS, and also most other *nix systems, such as BSD, and distros designed around the Linux kernel. This allows me to implement cross-platform support, for at least MacOS. Another aspect of Python that makes it highly useful is the large range of libraries available for usage with the base modules. For instance, I will be using the Pygame library for my project, but there are several highly polished graphical interface libraries available for usage with Python. In fact, there are libraries and modules able to fulfil most requirements for a given project, evidenced by the results of the 2020 Python Developers’ Survey, particularly the *What do you use Python for?* section. This decreases the time that one spends on constructing an inadequate solution, as opposed to utilising a function present in an existing library, which is both faster and typically higher quality. Because of the highly accessible structure, cross-compatibility and low resource cost of Python, and the wide variety of libraries, I firmly believe that creating my game in Python will be largely advantageous to the fluency and speed with which I develop it, and will allow me to produce a higher quality product.

System Specifications.

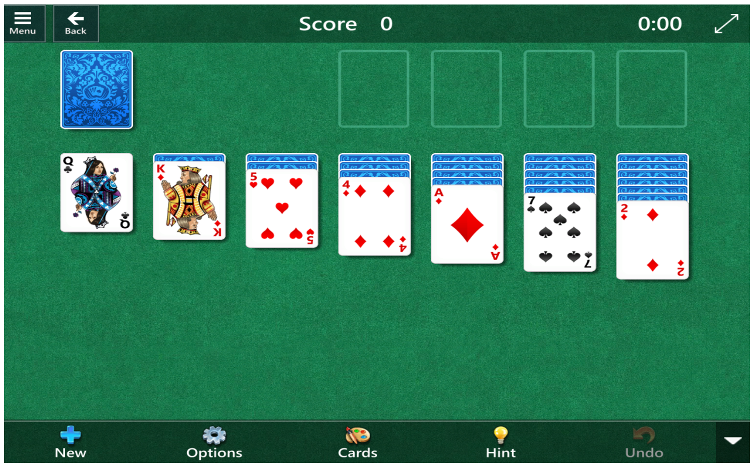
The game will be minimal in requirements, requiring only a display, a keyboard, and an ability to run Python. The ability to run on unix operating systems has not yet been determined, but it will most likely be compatible. The device will likely require at least 20MB of free space, in order to accomodate the assets used in the game. Any Intel CPU from at least 7th generation onwards should be able to run the game, with the ability to run SDL being a bare minimum. Speakers will be recommended for the best experience, since background music and sound effects will be included. These specifications allow the game to be accessible to many users, and can be justified by the functionality that my game must have. To graphically represent the cards in play, I have chosen to use pygame, and therefore a computer capable of running pygame, and, by extension, the ability to run Python and SDL, is the minimum requirement for my game's operation. From Pygame's *about* page:

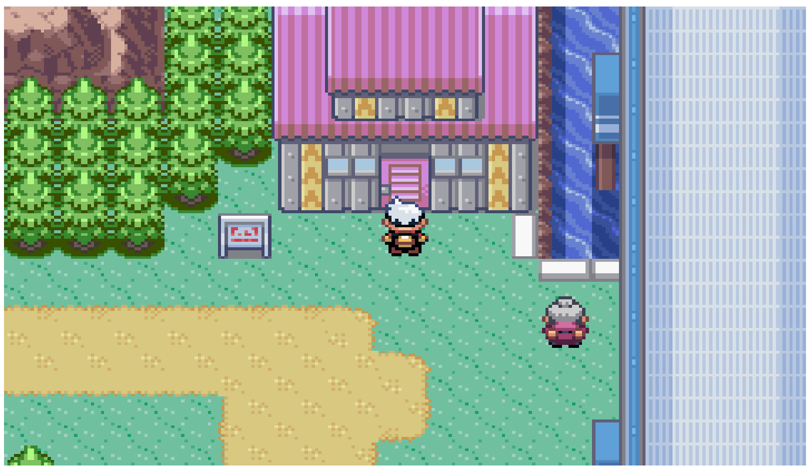
> "Supports Linux (pygame comes with most main-stream linux distributions), Windows (95, 98, ME, 2000, XP, Vista, 64-bit Windows, etc), Windows CE, BeOS, MacOS, Mac OS X, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX, and QNX."

By being able to run Python and SDL, most of my game's graphical requirements will be met, since these are the requirements for pygame. The requirement of keyboard and mouse input means that the user must have a keyboard and mouse, in order for input to be obtained. Since the requirements are simply the ability to run python and SDL, having a monitor, and, optionally, having speakers, my game's requirements are basic, and able to be supported by almost all modern computers.

Relation to Existing Systems.

My idea is inspired by several real-world examples, which utilise similar concepts. As a developer, my ideas are bound to be derivative of the works I have experienced. Most notably, I have been inspired by games involving cards, particularly those played with a playing deck, *Microsoft Solitaire Collection*, and Mattel's *Uno*. Some concepts I have come up with also derive from third-person RPGs, such as *Runescape*, or the *Pokémon* series. For this, there would be a world of at least one location, with at least three rooms, where the player would be able to move around and interact with different NPCs to play different games. In regards to multiplayer card elements, I draw inspiration from the Uno online game, which has impressive online multiplayer elements, and supports several variations of the Uno game. The visual representation is also derivative of Microsoft Solitaire, which has multiple solo games represented as being played on a large table. Instead of online multiplayer, my game will most likely feature pass-and-play functionality, and, if time constraints allow, and if I can get one to work, a computer enemy feature. In summary, I will be using a pass-and-play version of the multiplayer found in the online Uno game, the top-down perspective and location composition of older Pokemon games and Runescape, and almost directly mimicking the rules and gameplay of several playing card games, such as Poker, Blackjack, and Bluff, but will add a mathematical modification, such as changing poker hands to reflect some mathematical operator, making Blackjack multiplicative, or requiring a changing sum of cards to be played for a legal turn of Bluff. This will make the game seem familiar, but add originality to the execution. Therefore, these are the games that I have used to draw inspiration from.





Approach Towards Development of my Project.

I will develop this project using the prototyping approach, combining rapid throwaway and evolutionary prototyping. Since I do not have much experience with pygame, I must learn how to apply pygame functions to a variety of scenarios, and therefore rapid throwaway prototyping can grant me the necessary experience to be able to construct solutions if the requirement arises. The rapid throwaway prototyping strategy also allows me to pivot my concepts and remain flexible in the development of concepts for my game. To construct the game itself, I will use evolutionary prototyping, adding onto my game as I develop it. By combining these two approaches, I believe I will obtain a resultant product that will effectively fulfil the requirements.

My decision to use prototyping is also driven by my lack of resources, and my solitude in the development of the product, wherein I lack substantial funds to hire external aid, and also lack the time required for a higher level approach, such as agile or structured development, which can take upwards of a year to produce a quality product. Since my target audience will require a functional product, which is based on the mathematical aspects of gameplay, rather than the aesthetics and refinement, I can afford to have a final product that is not of commercial quality. However, since this will be developed for others, the end user approach would not be suitable. Furthermore, I will require flexibility when constructing my game, as I am working alone, and may decide to pursue an alternative direction in certain aspects of the game, requiring me to change substantial sections of my plan and code, in order to ensure forward compatibility. For these reasons, I have chosen to utilise the prototyping approach, combining rapid throwaway and evolutionary prototyping.

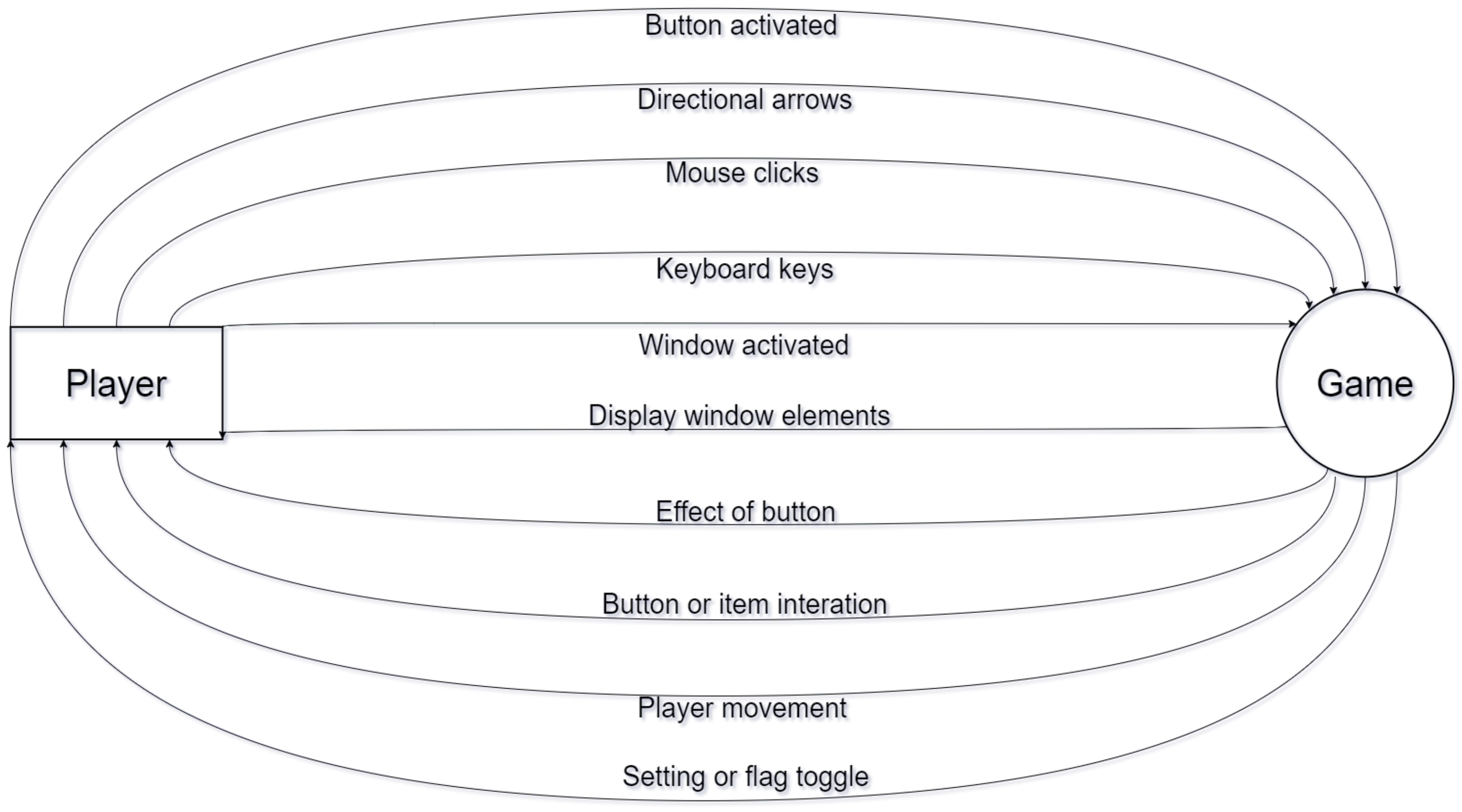
Documentation of the System.

IPO Diagram

Input	Process	Output
Arrow key (left, right, up, down)	Check arrow key input	
	Call controls(), with arguments for the direction(s)	
	Change the player sprite's coordinate attribute in the direction by the speed value	
	Update player icon and position via place()	Player movement
Mouse click/input	Test for mouse input	
	Test for if in game	
	If in game, continue, else end	possible exit
	Get cursor coordinates	
	Compare against rect of currently present buttons	
	Activate button if clicked	Button press
Activation of Window	Check window size parameters	
	Create window of specified size	Black Window
	LOOP: main_game_loop()	Window with elements
Button toggled	Check button identifier	
	Toggle button ID flag	
	Change Button sprite	Display button on sprite
Other keyboard keys	Check Key Code pressed	
	If key [item]: trigger (item)	
	If key [ability]: trigger (ability)	

(item)	If item in range: trigger item.effect()	
	Change item sprite	Display item use sprite
(ability)	Toggle flag Ability	
	Ability.effect()	
	Change sprite for ability	Display ability sprite
	Toggle flag to 0	
	Revert sprite to default	Display normal sprite

Context Diagram



Quality Assurance and Assessment.

Once completed, my game will have to be assessed in order to ensure that it is at the required standard of quality to be utilised for its intended purpose (i.e. to operate as a mathematical card game compilation). This is a crucial step in the development process.

Viability

My product will have to appeal to the target market (i.e. those who enjoy mathematics and cards), in order to be considered successful. If my game is made poorly or improperly implements the mathematical inflections, it will not have an appeal and therefore will fail to meet the requirements. To ensure that the project is appealing and fulfils the requirements, I will conduct user research, to better understand the evolving requirements of my target audience. As I develop my game, I will have to adapt the functionality to these evolving requirements, in order to ensure the continued appeal of the game. Once I have a low fidelity prototype, I can directly test user responses to the project, allowing required changes to be made, as per user feedback. Once I am in the final stages of development, I will employ User Acceptance Testing with at least five users, providing me with an overall measure of the projected success of my product, and the alterations that must be made to maximise the appeal of the game to mathematicians who wish to play mathematical card games.

Usability

Following this, the product must also be user-friendly. This means that existing conventions for graphical elements should be followed, for instance using circles for radio buttons, or having a visual indicator for button states. Since visual elements can be ambiguous, to avoid the necessity of redefining the meaning of visual element shapes, using colours and shapes used to indicate concepts in other games, such as having a red cross for a cancel button and green tick for an accept button. The keyboard controls should also mimic those of other games, for instance using the escape key to activate the menu and pause the game. By following this, I can ensure that users will understand the functionality of my game. To further reinforce this, I should have a tutorial or some instructions page of some kind, in order to inform users of the controls where they may be unintuitive, such as if the interaction button is bound to “g”, or if using the third ability is “d”. By following these guidelines, I can be mostly certain that users will be able to understand and effectively play my game. Compounding this, if an error does occur, the error must not completely crash the game without an indication, rather it should trigger an error flag and then exit the game with an error message. This error message should be readable, explaining what error occurred, using language that a user can either understand or easily search up. For testing purposes, a more verbose error logging feature can be implemented, having a toggle between verbosity and simplicity.

Efficiency

The code will have to be efficient in order to function properly, and not unnecessarily slow down the device of the user, or unduly limit the hardware capable of running my game. This will require the utilisation of optimised code, ensuring that the processes occurring are necessary and constructive, rather than clobbering. My code will also have to be easy to maintain, if bugs are discovered, or some functionality is impaired due to incorrect structures or logic. Therefore, I should accommodate this in my designing of the code, making the code modular, with the use of classes and functions, and by adding many comments, in order to make my thought process easy to follow and correct if required. If, for instance, colliding with a certain NPC in room 3 causes an unintended event to be triggered, it would be much easier to locate this error in my code if each room and sprite is easy to locate, being sorted and modularly placed, and if each room was loaded as a separate loop, rather than all being loaded at once. My code should also accommodate future alterations to base functions and values, using variables instead of absolute values, and functions instead of builtins. Finally, the game must be portable and able to be run without much preparation, and preferably should be able to run on operating systems other than Windows. I have the capability to test this for, at the very least, MacOS, and ensure running.

Testability

The game must therefore be easy to test, for both errors and smooth operation. This can be done by making the code modular, with functions for each loop and event handling, such as having a main game loop function, which calls each room's loop within, allowing reordering and restructuring of the behaviour experienced when moving between locations, and when an event is detected. This would greatly improve the speed at which I debug the game and implement new functionality, and would also provide significant forward-compatibility. Another feature that would greatly aid me in testing inputs would be breakpoints, which would allow me to understand the flow of data at a given point in the execution of the code. This testing would also be conducted by members of the target audience, as outlined previously.

Standard and Common Functions

My game will undoubtedly utilise many functions that are standard or common to many Python projects, including those I have previously worked on. For instance, a main while true loop is essential to all pygame scripts, constantly updating the screen, and sorting and searching algorithms are standard to Python, with many being built-in. For the purposes of working with coordinates, the Complex() datatype can act as a bridge between polar and cartesian coordinates, making it ideal for vector calculations, with modulus-argument notation. By extension, these vectors can be used in the movement of elements on the screen, which is another feature common to pygame projects. Many common functions can be found online, on various forum sites, such as Stackoverflow. By adapting code found on these sites to my own purposes, I can save time and energy, and use more efficient functions. This can also contribute to the modularity of my code, making it easier to organise, and read it. Thus, by reusing existing scripts, I will save time and produce a product of higher quality, since I will also have more time for refinement and user testing.

Project Plan

To fully ensure that I do not stray from the time constraints and project requirements, a plan should be implemented to guide me in the development of the game. This should include a Gantt chart for a schedule, a data dictionary for the organisation of data structures that will be used, and a clear concept of the success metrics. The use of a Gantt chart will visually represent the stage of development that I should be up to in my coding as I progress, and help me to pace myself, and allocate time effectively. A data dictionary will aid me in organising my ideas for the variables and data stores that will be included and utilised in the project, such as scores, player, and card classes, with a base element class, and other variables which will be used. The usage of success metrics will help me understand whether the final product truly meets the expectations of quality and functionality. By having a clear goal to fulfil, I will be able to plan towards the successful fulfilment of requirements. Thus, by implementing a project plan, with these elements in mind, I should be able to have an organised and structured approach to the time and direction of the project.
Complimenting this, I can also use a logbook to track my progress, so I can understand the work that has been completed, and that must still be completed. This will also allow me to record my thoughts and understand the code I have written, as well as allowing me to cite references for code, so that I can use these sources in future, and also credit these sources as inspirations. These strategies will allow me to maximise the quality of my work.

Summary

Thus concludes my major project proposal. I will be using Python to make a card game with mathematical emphasis, inspired by multiple real-world games. I will be using a prototyping development approach and implementing multiple strategies to ensure the quality of the final product.