 Game Project Write-up Template

**1. Describe the game you wish to develop:**(Objectives, constraints, success criteria, reward system)

|  |
| --- |
| E.g. 1. My app is a game titled ‘Cannon’. I wish to develop a side view projectile-motion game, where a player can choose an angle, and choose the strength, to launch a projectile to hit a target. The player will have 5 tries to hit the target. The target is a 5 x 5 metre square, and it will be 80m away. There is a 10m high wall in-between the player and the target, The range of angles that can be thrown will be from 10 degrees to 80 degrees. The strength will be from 1 to 30m/s. Since I plan to use pygame to create a Graphical Display, I need to define the window size as 800x 400 pixels. A suitable scale will be to use 800 pixels to represent 100m.  E.g. 2. Number Guessing Game. The computer will guess a number which is between 1 to 100, and the player will have to guess it. The computer will give a response whether the number guessed is ‘cold’ which is very far away from the answer, or ‘warm/hot’ which is very close to the answer, or ‘correct’. If the player took more than 10 guesses the player loses. It will have a high-score table which shows who took the least amount of guesses. This is a flow chart of my program… [Start]-[Get user input]-  E.g. 3. The name of my app is called ‘Story Time!’ This is a game where a whole class of students create a story. The first 2 lines of the story are visible to the 1st player. He creates the 3rd line of the story. It will then be saved as a text file. Next, the 2nd and 3rd line will then be visible to the 2nd player, who will continue the story by creating the 4th line. This process repeats itself until say the 20th line where the game stops, and everyone contributed a line to the story. The whole story will then be presented for viewing. Laughter could possibly result. The names of the author of the individual lines could also be revealed for a good laugh.  E.g. 4. A two-player game, using paddles to bounce a ball left and right. The one who concedes a ball on his side loses… (...details … )  E.g. 5. The classic game asteroids |

**2. Describe the key data types/algorithms you will use:**(Data types, Inputs, Processes, Outputs, Loops)

|  |
| --- |
| E.g. 1. Floating point numbers like x, y coordinate of the projectile, and the x, y velocity of the projectile. Gravity will be the constant 9.8m/s. Inputs will be text based. I will need a while loop to simulate until the projectile hits the ground or goes out of screen.  E.g. 2. A list of integers, to store past guesses. This can be used to remind the player that the number was tried already, so that the player can choose another number.  E.g. 3. A list of objects. I will be using OOP to model the various objects in my Role-Playing-Game. The objects are Weapons, Player, and Food.  E.g. 4. I will use a relational database SQL to store data about… My tables will be…  E.g. 5. A list of string, to store the lines that players key in. |

**3. Describe what modules you will need to install, to run your program, along with instructions:**  
Warning: Do not use software development helpers like Visual Studio to do this project. Just use Python, it’s IDLE interface, and install python libraries to do this project.

|  |
| --- |
| E.g. 1. The python module/library pygame is required. Instructions to install can be found at https://www.pygame.org/wiki/GettingStarted What I did to install: In Windows, Run the Command Prompt as Administrator, and type “py -m pip install -U pygame --user” to install pygame.  E.g. 2. No additional modules are needed. It is a text based game, that does file reading/writing. It can be played in python shell screen. |

**4. Explain the software versions that you have submitted.**  
(Periodically save a new version, after you achieve some success/milestones in coding.)  
(Place comments in your code to explain what it is doing as well.)

|  |
| --- |
| E.g. Ver 1: Tested the user input and displayed the projectile at the start successfully. Ver 2: The projectile moves, but in the wrong way. But it works. Ver 3: The projectile can finally hit the target, but it does not seem to register the hit. Ver 3a: Made the animation slower to allow player to see the projectile animate. Ver 4: The projectile was allowed to go through the wall, but now this bug is fixed. Ver 5: The program loop is now correct, allowing the player to try 5 times before reporting that the player loses. |

**5. Any reflections on your final submission?**(Fill this up to document the interesting things worth highlighting in your journey to create an app.)

|  |
| --- |
| E.g. I didn’t know pygame can do \_\_\_\_\_\_\_\_\_. I learnt what is a game loop, how to use timer, and how to control a frame, and frame rate... Pygame has many more features that I didn’t explore. Given more time, the display of my game could have been more interesting, by using a picture of a robot as a target, and a cannon for the player. Perhaps I could have tried using pygame functions to tilt the cannon so that it points in the same direction as the angle of inclination, before animating the bullet.  After completing the game, I felt it was too easy to win. It could be so much more interesting and challenging to allow the robot move around every turn, instead of staying still. |

Rubrics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rubrics | Exceeding Expectiations | Meeting Expectations | Approaching Expectations | No Evidence |
| Documentation | Documentation is complete and reflects game project submitted. | Documentation is mostly complete and generally reflective of the submitted project. | Documentation is filled out, but is not complete, or it does not reflect the submitted project. | Document is illogical or missing. |
| Commenting | Code uses white space, good naming conventions, indentations and comments, to make code easily readable. | The program code makes use of white space, indentation, and comments. | The program code has few comments and does not consistently use formatting such as white space or indentation. | The program code does not contain comments and is difficult to read. |
| Programming | Effective use of modular functions/abstraction to organise code in logical segments. Iteration/Recursion is used appropriately. Program executes flawlessly. Sections not implemented yet is handled appropriately, and documented. | Some functions used to organise code in logical segments. Minor problems in execution: reporting wrong values, graphical imperfections, illogical/unexplained flow. | At least one function is in program.  Program has bugs/hangs/illogical flow. | No functions or modularity used in program.  Code does not function. |
| User Experience | Well designed, documented, and intuitive. A pleasure to use, with a smooth flow, from start to end. Instructions are shown where necessary.  Data is validated on input, and request to re-enter if invalid. | Meaningful experience with the program. May be too much instructions that fill up too much of the screen, or too little instructions. Inputs are difficult to key in, due to being invalid (unless they are strictly lower or upper case) or lengthy. | Some situations where user is lost as to what to do, and has to make a guess what input should be made. Apparently clickable objects on screen appear to do nothing when clicked on.  Invalid data leads to display anomalies. | User needs to read your whole code to try to understand what is going on, and how to interact with your program. |

Some IDEAS :

|  |
| --- |
| 1. Letter drop. (Easy)  Task 1a. Display random letters from A-Z, to drop from the top of the screen to the bottom.  Task 1b. New letter appears every 2-5 seconds. (120-300 frames if using 60fps)  Task 1c. If you type the correct letter, the dropping letter disappears.  Task 1d. If the letter drops (touches the bottom of screen), you lose.  Task 1e. Create a score display, that keeps increasing as you survive.  Task 1f. Allow input for a name, if a high score is set.  Task 1g. Add boom.mp3 sound for correct letter, and dumb.mp3 sound for wrong letter pressed.  Task 1h. Increase difficulty over time by shortening the interval between letters.  Task 1i. Decorate the letters with squares/circles around them  2. Letter shooter.(Medium)  Same as above, but using a tank to shoot bullets at letters.  Task 2a. Letters don't just move directly down. They move diagonally down, bouncing at the side of the screen.  Task 2b. Letters go down faster due to gravity.  Task 2c. Tank can shoot projectiles in 2 second intervals  Task 2d. Tank can shoot 2 projectiles at a time.  3. Sniper (Medium)  Task 3a. Animate a bird flying left to right  Task 3b. Create a yellow target-scope as your mouse cursor  Task 3c. When the target-scope is near the bird, it changes to red  Task 3d. When you click to shoot and hit the bird, it drops down screen and you get points.  Task 3e. Bird flies in a bird-like path, making it more difficult to aim.  Task 3f. Set shooting delay to 4 seconds to make it more difficult.  Task 3g. Insert shooting sound. Insert bird dying sound.  4. Driving (Easy)  Task 4a. Animate a (top view) car at bottom of window that faces up and can shift left to right  Task 4b. Create a random size rectangle that drops from the top of the screen to the bottom.  Task 4c. If a rectangle hits car, you lose.  Task 4d. Create a list of rectangles that link together from top of screen to bottom.  Task 4e. Create a duplicate set of rectangles parallel to this list to simulate a road.  Task 4f. Create an increasing score as player drives along road. Allow up/down acceleration  to change the speed of car and score.  Task 4g. When car crashes, and a high score is met, allow entering name.  Task 4h. Road becomes more narrow as game progresses.  6. Lunar Lander (Difficult) Task 6a. Create a lander by blitting it on screen, and making it move UDLR. Task 6b. Instead of controlling displacement, the movement buttons now control acceleration. The velocity controls the displacement. Task 6c. Introduce gravity into the model. Task 6d. Make it detect when it goes off screen.  Task 6e. Check the horizontal velocity when it goes off screen sideways. If it is too big, player loses. If it is small, change the velocity to make it appear to bounce back into screen. Task 6f. Do the above for the vertical direction. |