# Project Post-Mortem

Part 1 (2-3):

Briefly describe your additions to the project. This is also your opportunity to describe everything that you did even if it didn’t make it into your final submission. In short, justify your time spent on this project.

The aspect of the project that I am most proud of is my modified A star pathfinding algorithm, which allows enemy sprites to platform and keep up with the player. My algorithm also searches every node on the screen rather than the entire map, which makes searching more efficient. I am also proud of the fact that with a click, the mouse position can be added to the camera offset to initiate a target vector, at which point the target and the player’s position are normalized to create a unit vector that guides the player’s ax to the target. These creative additions were the most challenging and time-consuming to implement.

Part 2:

Breakdown each of the additions you have made as they fall into the project description categories. Also include briefly where/how each is completed (you may list lines of code line 315 to line 380, or classes Player class, or individual methods Player::Move() to justify your points). After listing where it can be found also give your own assessment of the points you’ve earned. I will then provide my own assessment when grading.

<See table on the next page>

| **Points available** | **Where to find in your project (Traceability)** | **Your**  **Assessment** | **My Assessment** | **Category description** |
| --- | --- | --- | --- | --- |
| 10 to 40 | Filename: PathFinder  This class creates a graph of the entire map and then use a path finding algo to find the best path to the player  Filename: StrikerEnemy  Filename: TurretEnemy  They are created based off of their given id in the Tiles class. So if I put a 3 in one of the spots in a my map.txt the World Handler will draw the that specific enemy at that location | 40 |  | **Enemies** should be other entities within the level that try to impede the player’s progress. They may be set at a current position in the level every time or they may be spawned entirely randomly. They can also just ignore the player and go about their business or they can actively pursue and try to defeat/stop the player. Points up to 30 are given based on the complexity of the enemies (do they have sprites? How are they created? Are they hard-coded (bad don’t do that), do they have any AI/intelligence?) |
| 5 to 15 | FileName: StartMenuState-render()  This is where the tutorial is drawn to the screen | 5 |  | **Tutorial** – Show a tutorial on how to play your game (a few points for written directions, more for actually teaching players how to play the game). Note that if your really ambitious there is the Robot class which can take control of they input devices and move the player through code to demonstrate different skills and abilities. |
| 20 | Image or word file (do not include a file format I have to upload to open or NO credit will be given) | 10 |  | **class diagram** – including attributes, behaviors and associations between classes. Ideally, the class diagram should be done prior to working on the project and in such case the final project does not have to match the design perfectly but some consideration should be taken to update the diagram if the project drastically deviates from the design. |
| 10 to 20 | Filename: Player-canAttack()  Keeps track of the system time and only lets the player attack so often  Filename: PlayerAxe  Player now has a ranged weapon. | 20 |  | **Player improvements** – The player can already run and jump around the level. Improvements may be additional skills (shooting enemies, or wall jumping) or general code improvements. |
| 5 to 10 | Filename: Player, Sprite-die()  After an enemy sprite is killed the player gets a point | 10 |  | **Scoring** – each game should have some kind of score (increased when floating apples are collected or shiny pink stars). Scoring should be displayed to the player(s) as the game continues and should be shown to the player as the game ends (possibly being augmented by the time it took to complete the level). It should also be kept track of when saving and loading the game. |
| 5 to 10 | Filename: PauseState  When the player pressed P on the key board this menu pops up. It lets the player save, return to the main menu or resume the game. | 10 |  | **Out of game menu**. It does not have to be elaborate but some kind of opening screen when the game starts where the user can decide to start a new game, load an existing saved game (if you’ve done saved games), access options (if you have options), see the highscore list (if your game has scores) access credits (get noticed), and exit the game. At the end of the game, it should return to this start game screen. |
| 10 to 30 | Filename: WorldHandler-loadLevel()  Loads and creates the map from a text file. Each tile type has an id so this method know what to draw and where.  Filename: Tiles  This holds all the tile types and what they can do. | 30 |  | **Levels** – Have a variety of different obstacles and traps: points here will be given for different mechanics that these obstacles offer the player. Falling platforms, elevators, doors that need keys to open, pipes that warp you to a new level… |
| 10 to 20 | Filename: PathFinder-createNodes(), setupHashTable(), findPath()  Filename: PlayerAxe-move(),  Filename: Vector2D-normalize() | 20 |  | **Code** – your code should be easily readable (Object Oriented Principles followed, well formatted code file, comments where appropriate, variable names that fit their use). Algorithms should not be copy-and-pasted 20 times, and main methods should not have a 1000 lines of code etc... |
| 5 | N/A |  |  | **Cheats** – making cheats that allow you to pass through sections of the game to test/see the rest of the game. |
| 10 to 20 | Filename: Utility-writeToFile(), loadFile(),  The load file methods loads a map.txt into a string so it can be read by loadLevel() in the WorldHandler.    Filename: WorldHandler-determineLevel()  Looks at what btn was pressed on the start menu and if the load button was pressed check to see if a save file exists and if so load the file and start the player there. | 20 |  | **Serialization** (saving and loading the game) – The player should be able to pause and save and load the game over again. (or optionally access loading a game from the start game screen). |
| 5 to 10 | Preferably as an AVI, MOV, or MP4 |  |  | **Video** – make a video of the game and all/most of the main features (I’m not asking for a hype or teaser video but a playthrough) to demo the project. |
| 0 to 150 |  | 150 |  | Total is 200 possible points, but the most bonus points you can earn is 50 points. |