Revised Product Backlog

| **Feature** | **Story** | **Initial**  **priority** | **Task** | **Implementation Notes** |
| --- | --- | --- | --- | --- |
| Implementation of Trains | This is the central part of the game, which involves the user having to dodge trains while standing on train tracks. The game will require an interactive window. | High | Create a window that displays all the game compartments on the screen. | The dimensions of the game are set and the background of the game is set in a solid black colour |
| Create a train class and use the train algorithm that generated the tracks | Using the same algorithm as the tracks, the trains spawning are randomised except for the first trains which never spawn on the first track |
| The screen will be completely dark other than the area around the player and at the front of the trains. (as they drive down the  track) | High | Create a shading  class that shades the screen of black in  areas you want. | Make the background dark and just draw the parts of the train that should be visible which is the front part of the train |
| Create the trains moving down and keeping the light around them. | The train moves at a constant speed down and is redrawn each time it moves |

| Implementation of health  Plane Implementation | Every time  character is hit by a train/jumps onto a moving train, their mental health will decrease until “0” (from stress) | High | Create a counter  inside the character’s class. Keep track of the number of hits and display it on screen. | Instead created a health variable that starts at 100 and decreased by 20 each time |
| --- | --- | --- | --- | --- |
| During the planning period, random airplanes will fly on the screen containing  information on where trains will be during the next round. When they are initially flying,  they will glow, but the glow will slowly disappear as they fly. | Med | Use the algorithm  from the tracks and trains to generate the planes that have hit boxes to allow the  player to click on them to generate  hints for the next round. | Same algorithms used to implement the planes spawning but a maximum of three is spawned as any more hints would make the game less visible. |
| The planes will contain information such as:   1. What track will a train be on at what time 2. What the busiest track will be 3. The safest track for the round | High | Has a database of hints of usefulness or not that is  generated by the algorithm and displays as many as the planes were clicked. | There is a list of 6 hints that are predetermined that are somewhat useless but 1 or 2 are helpful for the player to avoid trains for the rounds. |
| The user will be able to click on the plane while it is flying to gather the information for the next round | Med | The click will increase the counter of  information given | When the plane is clicked the hint is displayed on the left of the screen and the plane disappears. |
| Planning period | During this period the user will get hints before the trains start heading towards the player. | High | Have the timer reset to 30 when it hits 0, which halts the signal of generating the new trains till the timer hits 0 again. | The timer is put on the middle of the screen and counts down. When it reaches 0 it moves on to the game phase. |
| Create a band of hobos | We want the user to have the ability to befriend the hobos and create a “band of hobos,” that work together in dodging the trains. | Low | Have a friendship  counter on the hobos, and once the hobo’s friendship reaches a certain level, it will follow you and help you | After a few rounds the hobos will start following the player around and mimicking the movements the player makes |
| Visible Hobos | The same hobos  that send the planes will also be visible on the track trying to dodge trains. | Low | Have a generated  hobo that moves in a randomised pattern on the track. | The hobos are drawn as 3 hobos and move to the left until it can’t anymore and goes on to switch to moving to the right until it can’t anymore and it repeats until the round is over. |
| Tracks on  the screen | This will be the game's playing field, as the player-controlled character can only jump between these. | High | Create an algorithm that generates the tracks randomly on start-up. Draw the tracks using the algorithm. | Use pygame random library to generate random integers to generate the number of tracks randomly → this is now changed for tracks to be consistently be 4 → 5 lanes |
| Weather | We want things  falling constantly to add intensity and atmosphere. | Low | 1. Create the different types of weather  effects. | Added rain drop images for rain and put white circles for snow |
| 2. Have a randomly generated weather effect for hail, snow, and rain. | Use the python random library to decide if it should be raining or snowing |
| 3. Have brimstone weather only when the user is one train hit from losing. | When hp is equal to 20 if it is raining it switches over to snow and starts falling much faster than before. If it was snowing before it falls even faster. |