RED = currently working on

YELLOW = partially done

GREEN = done

CYAN = maybe will do

Menu with maybe a shop for cosmetics as to reward the player for playing the game?. And play button, can also be used for pausing game because single player

Background design (with a sun rising and falling to show timer left for certain difficulties)

Disappearing platforms/moving

Platforms that detect player presence (button/plate), as well as collideables

Health + Lives, link with consumables so health and lives can be regained

* Difficulties would be nice and quick to add
  + Decrease number of lives
  + Add a damage multiplier (take more damage the higher the difficulty so players have to think more about how to preserve health)
  + Add a speed multiplier (slower the higher the difficulty)

Win screen to indicate to the player that they beat the level/ loss screen if they ran out of lives

Respawn system for every life used, if their health falls below 0, respawn them from last checkpoint (level)

Make a functioning boss that faces the direction of the player depending on what side of the screen they are on, that can attack, defend etc

Tutorial teaching player controls, etc

* Defence controls e.g F to block – provides block frames, parry frames within first 0.22 milliseconds
* Q to dodge – provides invincibility frames

Settings, and timers

*Static obstacles*

*Trap/fake static obstacles*

*Multiple levels increasing in map difficulty*

*Collectables/obtainable items*

*Controllable player*

* *Do a space bar jump control, where players gain height while gravity variable continuously makes their velocity decrease until they are falling, fall until hit a boundary e.g platform or void.*
* *Run button, just increases X axis speed temporarily for a certain amount of time until stamina runs out*
* *Make the character itself and draw it in the game*

*Loading screen with skip button and tips*

*Errors numbers:*

1. *Collisions – if moving while above or below the platform, my character gets teleported to the side of the platform, i.e if on top and moving left, I get snapped to the right top corner, if on top moving right, snapped to top left corner, if bottom moving left, snapped to right bottom corner, and if bottom moving right, snapped to bottom left corner. This is because when I am on the top or bottom of the platform, the game’s horizontal collision system kicks in. Also if I haven’t moved vertically recently, the character can just walk right through the platform from left to right or vice versa because I think the code updates all the x and y attributes of my character’s rectangle first before running the part of the code that checks x and y axes’ collisions*
   1. *To fix the first issue, I refreshed the variable that makes a list of objects from a grouped set of sprites (my level’s platforms) that my character collided with, this way, my character’s collision detection is always using the most up to date information on my character’s position, so it knows that I am on top of or under, or to the side of the platforms, the issue wasn’t in the code that does the collision detection, but rather with the information the code is fed.*
   2. *To fix the second issue I made it so that vertical and horizontal collisions and movement are dealt with completely separately, this way the code doesn’t get tangled and confused with the positioning of my player. I made horizontal collision detection first so that before the update() method starts detecting collisions, it updates the player’s rectangular hitbox’s position, then detects collisions. After X axis collision detection, I then implement vertical movement, adding gravity first then updating the player’s rectangular hitbox y position by adding on the updated y axis velocity.*
2. *Collision inconsistencies – my character occasionally, when standing on the platform will fall through it, and fall through very quickly due to vertical velocity accumulating (since I haven’t yet found a way to make it so gravity doesn’t get added when I’m standing on something).*
   1. *To fix this issue, I will try to instead of making collisions exact, give the platforms a bit of an extra layer so that the collision wall isn’t so thin, so if even 1 pixel of the character crosses into the platform, the collision script detects.*
   2. *The reason this is happening is likely because my addgravity method for my character is always on, when it should only happen when I am in the air or jumping etc. so when I’m standing on something it doesn’t add, this causes my velocity in the y direction to continuously get larger and larger to the point where it gets so large the code fails and I fall through, this is because my frames per second set for the game is 60, with this in mind the gravity adding method will add to my y velocity very quickly, causing this issue especially with thinner platforms I stand on.*
      1. *I fixed this error accidentally when I was adding terminal velocity, since I don’t want the player to fall too fast if they jump down from high altitudes, I added this, but this also answers b because by making it so the velocity vertically is capped to some maximum value, that also means it cannot become ridiculously big and therefore cause the math in the code to fail.*
      2. *Along with this I also made it so that I can only jump if I am touching the ground by making it so after collision checks are done, if I am detecting to collider with an object and am on top of it, I will be able to jump, otherwise I cant*
3. *World scroll – the scroll works, it does its job but all the platforms’ hitboxes are out of place, for example, the map is drawn a bit to the right while the map’s platforms’ hitboxes are all to the left of the drawing, making the collisions inaccurate because my player is colliding in places it shouldn’t be.*
   1. *To fix this, I made the offsets also affect the blit on the player, this is because originally, if the player moves, the player moves on the screen and the objects also move around relatively, but it is very inaccurate, the initial idea was to have the character be centered, and only everything around it is moved relative to the character’s movement. By having an offset variable that is subtracted from the x and y positions of each object, and the x and y position of the character, the character can I stay in the centre, while everything else moves if I move my character*