OCR GCE A
Computer Science NEA Project
Name : Ali Ali Mostafa
Candidate Number : 7022
Centre Number : 12332
School : Kingsbury High School
Title Of Project : Red Hood Runner

Analysis

Introduction To My Project

- The game I have chosen to design will be a 2D Platformer design where there will be one main character which the user will control though various levels. The game will be set in a vibrant world with enemies and obstacles. The key to passing through the levels is to jump between platforms avoiding hazards. Some features the game may include are: engaging levels, character abilities, dynamic environment, enemies and hazards, collectibles and maybe power-ups.

Target Audience

- This game appeals to a very diverse audience as young kids will find it fun and challenging and even people in their 20’s and 30’s even up to 60’s can play it to relax to detach from their life and problems. Mainly however younger players will be attracted to the colourful graphics and simple gameplay and other people may find the engaging levels creative and enjoy it. To be more specific, casual gamers may be interested as they would like to gradual increase in difficulty throughout the levels, Retro game enthusiasts may like the classic games style giving them a feeling of nostalgia and letting them relive childhood memories. Many families may have a desktop computer or a laptop allowing for younger kids to play the game

Identifying Stakeholders

- My game is a platformer game aimed for young children to teenagers as it is simple which isn't very complicated meaning its easy to play however not very stimulating and so older adults may not enjoy playing it. A stakeholder is someone who has an interest in a particular organisation or project.

- My stakeholders will be friends of mine , Emir , Murtadha and my younger brother Hussain. Emir is 18 years old and used to play a lot of games when he was younger , however now he has gotten really busy with studying and only plays a little amount of time throughout the week, however he has good experience. Murtadha is also 17 years old and plays games regularly while studying for school. On the other hand i will also be interviewing my younger brother who is 13 years old and spends most of his time playing video games, this combination of stakeholders gives me a wide range of opinions as i will have someone who used to spend a lot of time playing games and knows the older gen games but now doesn't and someone who spends a lot of time playing games , especially the new gen games , so this gives me a variety of new ideas when i will interview them.

- I will be interviewing both of my stakeholders twice. The first interview will be a general questionnaire asking them a few questions , whereas the second questionnaire will be more specific questions based on my game. I will be giving a summary of each interview using the answers given to me

  1st Interview with Emir (general questions):

- I asked Emir the type of games he plays and it ranges from action packed games like 1st person shooters to quiet simple games such as minecraft. He prefers multiplayer games most of the time as he prefers to play with friends since it's more fun and competitive, but every now and then he does play some single player games to relax by himself. Emir likes to play on keyboard and mouse since he plays on pc and as he is used to it and he says he thinks it gives him advantage over other players who play on controller since he gets to use more keybinds and click more keys in a shorter amount of time. His favourite game is minecraft as it involves many different concepts such as fighting monsters to stay alive, building homes to sleep in, creating cities and hunger meaning you have to hunt for food by killing animals. He likes when games have many colours and look good, also when each level is something different and not repeating as this makes the game boring.He doesn't like the game's screen size being too big as for example, if his health bar is in the top left of the screen, he doesn't like having to move his entire head, he would rather have the screen size smaller and only have to move his eyes to look at the top left of the screen.

  1st Interview with Hussain (general questions):

- Hussain plays games which include a mission like completing a level or finishing a plan. Hussain also prefers multiplayer as he says there is more to the game when you play with others and playing by yourself is boring. Hussain's favourite game is also minecraft as he likes the fact that there are a variety of objectives to complete and he also likes creative mode where you can do whatever you want and you have access to everything within the game. He doesn't like his game's screen size being too big as he then can't focus properly since he has to keep looking at the corners of the game for important information such as the health bar or anything else. Hussain primarily uses an ipad to play and so he presses on the screen to use the controls, however he says that he thinks he would prefer mouse and keyboard as it will allow him to press more buttons in a shorter amount of time which would help him play better

  1st Interview with Murtadha (general questions):

- Murtadha enjoys shooter games , he often plays krunker and valorant, krunker is a free for all shooter pixel - style game which can easily run on a web browser, valorant is more intensive game where two teams go against each other to plant a bomb and the other team needs to defuse it in a certain amount of time. He usually plays games rated 13+ as games which under that age are more boring he says. He prefers multiplayer over singleplayer as he likes playing with friends since it's more fun. His favourite game is beamNG , he says he enjoys the physics of the game especially when crashing cars. He plays on PC and prefers using

keyboard and mouse, he doesn't like controller as he feels it's uncomfortable and doesn't let him play to his full potential.

2nd Interview with Emir (more specific):

- I asked Emir what are the main things that should be displayed on the main menu and he said that the play button and help button should be displayed as these are the most important buttons a player needs, if the player doesn't know how to play then he cant play the game and so on the help screen it needs to display the controls and the main objective of the game. I asked him if the levels should be able to be selected from the menu screen and he said no and that the player should be able to progress through the levels by completing each level. I asked him how long should a level take to complete and he said that it should be challenging and take maybe 1 -3 mins but it shouldn't be too hard as then the user may get discouraged and not want to play anymore. I asked him if i should change the blocks the player is using to move around the map with each level and he said that i should as this will bring more life to the game and make it more interesting and stimulating. He said it will also make each level unique as it is completely different from the one before. He also mentioned that he would like to see different characters for each level as this will make the game look even better. I asked him if the background should be different for each level as well since the blocks are different and he said definitely since this is something that the user lays his eyes in the very first seconds of playing the game and so by having different backgrounds, this will make the game look much better

2nd Interview with Hussain:

- I asked hussain what should be on the main menu screen and he responded that that first screen a user sees should be vibrant and grab the user’s attention as this will dictate whether the user has a good or bad impression of the game and it should have a button to start playing and a button quit incase the game was started accidentally. I asked him if the levels should be available to access and he said yes since if a player has already completed for example, level 1 and 2 but then stopped playing , when the player comes back he would want to continue playing on level 3 and if there wasn't access to the levels then he would have to complete level 1 and 2 again. I then also asked him if the background and blocks/tiles should be changed each level and he said that for the background yes since it will make the game more interesting and look better making the user more consumed within the game however for the tiles , they shouldn't be changed because that might confuse the user as each level the tiles change and so they would not know where to move the character

2nd interview with Murtadha:

- I asked Murtadha the minimum amount of levels that should be in my game and he said 2 - 3 but preferably three since if it's only two then the game will end really quickly. I also asked him how many obstacles should be implemented within each level and he said at least two main obstacles so that the level is kind of challenging but not too challenging. He also thinks that the main menu screen should have a play and help functionality and on the help screen there should be context of what the game is about and the objective of the game as well as

the controls. I asked if the user should be able to select what level they want to play on and he said no as then some users might just skip to level 3 which is similar to cheating and you should only be able to access level 3 after finishing level 1 and 2. I then asked him if the player should be able to double jump and he responded with that yes as it makes the game more fun and challenging however not on every level to make the game more interesting like only on one or two levels. Something that I was considering adding into my game was a monster which chases you as you try to complete the level and Murtadha said that that is a very good idea however try not to make it too fast and maybe add it only to the final level.

Analysis of interviews with stakeholders :

- From the first interviews with my stakeholders I have found that most of them like a game which has a main objective to complete, whether that be to eliminate the opponent team or to build a house to sleep in. Two of them play on PC which is good since my game will only be playable on keyboard and mouse. They all prefer multiplayer so this is something that i will need to consider since 2 player games are more fun to play especially with friends.

- From the second interview I can tell that all the stakeholders want the main menu screen to have a play, exit and help button. This allows for easy functionality within the game. The main menu will also need to be vibrant and eye - catching, this is to grab the user's attention and to make the game look good. This will be done by using bright colours and text to display the game. I have also decided that the user will not be able to select the level from the menu screen and instead they will have to complete levels 1 and 2 to reach level 3.

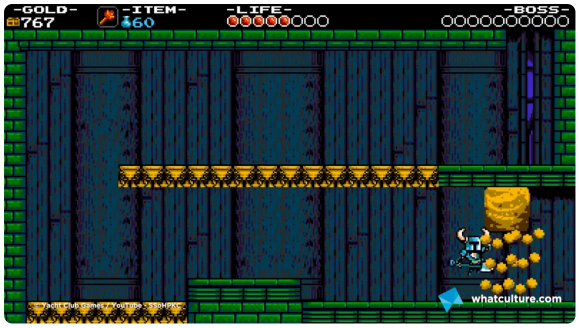
- I have decided that each level's background will be different, this will bring life to the game as well as making it much more visually appealing. Adding in a different background for each level allows me to set a different theme and vibe for each level, this makes the game more interesting and fun since every time a user finishes a level they can expect something new and different in the next level. I also do want to add different tiles/blocks within each level as this will add to the visual appeal of the game and enhance the effects of the different theme within each level however this is something which i have not finalised yet and will decide on in the future, This is because i do not know how hard it will be to implement it into the game and if i have enough time or not to implement it

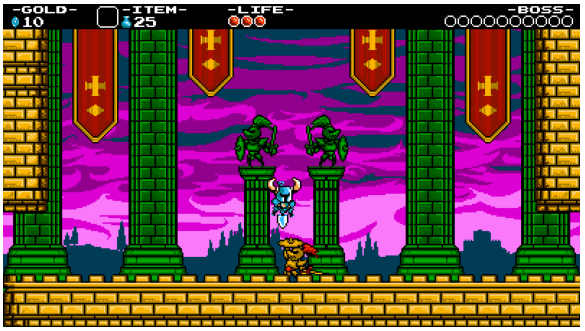
- The character will have the basic controls of moving left, right and jumping up. I have decided that I will not implement double jumping and only allow for a single jump.This is because if I allow for double jumping then the user might use it all the time to go through shortcuts within the level. I would want the user to go through the entire level without taking shortcuts and so only single jump will be allowed . I will also only have one character throughout the game as I do not have enough time to implement a new character for each level.

- I do want to add a monster that chases you within level 3. This is because it will make the game very fun and make the user have something to be waiting for at the end like a final boss they need to escape. However i am also not 100% sure I will be adding it in the game because i can imagine it will might be hard to implement and i do not currently know how i would implement something like this but if i do have time it will be something i want to add and i will definitely learn how to add it.

Detailed Research Into Similar Games

- Shovel Knight (2014)





- Brief Summary Of The Game

  Shovel Knight a 2d platformer game which has a retro design, it was released in 2014. The game is about a knight who uses a shovel as his weapon to take on bosses in fights to rescue a friend of his. It is set in a fantasy world. It uses 8 bit pixel art with modern game mechanics. It focuses on exploration and defeating bosses

- Weapon Mechanics

  The Knight's main weapon is his shovel, he can use the shovel to dig up treasure or break things. There is a famous move called the “shovel drop” which allows the player to bounce

off enemies and objects. This shovel makes the game more strategic making the user more engaged and making the user have to time the attacks

- Level Mechanics

  Each level has a unique vibe to it with different colours and themes, it also has new challenges and different obstacles in each level. In most level there hidden places which can be broken into using the shovel, these areas have treasure which can be dug up or other powers which can help the user in fights. The game allows for people who do not want to follow the main things to try out different paths and different ideas. There are checkpoints throughout the levels but a really nice feature is that if the user wants to they can break the checkpoint for treasure instead however now they need to continue to the next checkpoint to save their progress

- Power - Ups

  Collectibles include gold , gems and other things which can then be used to upgrade the knights armor or other abilities. Some examples include “Flare Wand” and “Phase Locket” , these help the user by making the knight stronger and making the visual of the game more stimulating. Some power ups are really needed for defeating some bosses. There is a lot of variation in the different power ups available allowing the user to make his own style of the knight and fight the bosses in their own way

- Controls And Movement

  Basic controls include moving the player left and right as well as jumping. Holding the attack button for an extended period of time allows the knight to swing his shovel forward to deal damage to the enemy. “Shovel Drop” is also an advanced movement which i talked about earlier

Super Mario Bros. (1985)

- Brief Summary of The Game

Super Mario Bros was created by Nintendo in 1985 for the NES (the NES is an 8-bit home video games console). It is a 2D platformer game where players control either a character named Mario, or his brother Luigi and the moan objective is to save a Princess named Toadstool from the villain named Bowser

- Control System

The Directional Pad (D-Pad) is used to move Mario left or right. A button allows Mario to jump and holding the B button down allows Mario to run faster and shoot fireballs when fire flower power-up was activated

- Level System

The game is made up of 8 worlds, each with 4 sub levels. These levels contain many different environments like overworlds, underground levels, water levels and castles. Every level's ending ends with Mario facing a fake Bowser up until the final level where fights with the real Bowser

- Power-Ups System

Super Mario Bros has a variety of power-ups including: Super Mushroom – this is where Mario turns into Super Mario letting him take extra hits from enemies, Fire Flower – when you collect this it allows you to shoot fireballs which can do a lot of damage from a far distance, Starman – makes you invincible for a temporary amount of time and last but not least 1-Up Mushroom – this gives an extra life.

- Lives System

Players start the game with a set number of lives and can get more lives by collecting 100 coins, finding 1-Up mushrooms or performing other actions within the game. Losing all your lives means Game Over.

- Additional Important Features

Each level ends with Marion jumping on a flagpole, enemies include Goombas, Koops, Piranha Plants and more and each having its own different behaviours and Warp Zones are hidden areas that allow players to skip to different worlds.

Features Of Proposed Solution

- First thing to be loaded once the game is started is a vibrant and colourful menu, this menu will contain the name of the game and the different controls to move throughout the game. This will include the help screen , leaderboard screen and how to leave the game. This will be put on the main menu for easy access so that the user will always know how to do whatever they want. I will have a transition door at the end of each level, once the character collides with this door they will be spawned into the next level and if there is no next level then the game will end

- From both Super Mario and Shovel Knight I want to have an animated character since different animations will bring life to the game making it more visually appealing and giving quick responses to the user making the game more engaging since characters will respond in various ways to falling, jumping and colliding with obstacles. This will be done through the use of sprite sheets, which are large images containing a sequence of frames for each animation. These frames will be cycled through to create the illusion of movement. Benefits of this are that the game becomes more dynamic and engaging as well as giving it a high-quality look, this also gives a better player experience and it also gives a clear state representation allowing the user to which state the character is in. The character will always spawn in a safe position within the level so that the user can have a look at the level and figure out how to go through all the hazards and find the transition door.

- I liked the variation through the levels in shovel knight and will definitely want to do something like that for my game like changing the terrain or the player throughout the levels. I want to do this because it will keep the player engaged as the visual of the game will be really appealing. I also liked the power up system of shovel knight however i will add power ups in a different way like for example if the player goes on a certain block it will give him super speed for a little while then fade away or if he finishes the level under a certain amount of time then in the next level he gains a different power for a little amount of time. There will be three levels for the user to complete with each level looking different to the one before through changing the background colour and changing the tiles the character walks on. This helps since it will give each level a unique feel and will make the user curious as to how the next level will look like. There will be tiles surrounding 4 sides of the game, these tiles will act as barriers to the character as they will not allow the user to exit the screen.

- I will be implementing a score system into my game so that the user can track how well they are doing. This will consist of a timer that increases until the user completes the final level. The objective is to complete all three levels in the shortest time possible. This will make the game more competitive as it will push the user to complete the game as quickly as possible and to find the best path within the levels to the transition door that is also the shortest path allowing them to move to the next level even faster.

- I will add a leaderboard system where the top 5 scores will be shown, this will also make the game feel more competitive. This helps in keeping the users who play the game engaged and will make it more fun since everyone will be trying to get the number one position on the leaderboard. Also I will be adding an input feature where the user will be able to type in their name or anything else they want which will be displayed next to their score within the leaderboard. This will make the user strive for completing the game with the best score as then the leaderboard will show that.

- Lastly i really liked the idea in shovel knight of how the user can choose whether to use the checkpoint to save progress or to break it for treasure, i want to incorporate this into my game but in a different way like for example if the user doesn't want to save his progress he can can obtain a superpower for a limited amount of time or he gets extra points on his score since if he loses on level 3 for example and did not use the level 2 checkpoint he has to restart from level 1.

Limitations Of Development

- I do want to add powers for the characters and also enemies to fight which could get progressively harder however these may not actually be in the final game as it may be too hard to implement. These powers could include an extra life for the user or powerups like being able to shoot with a gun however trying to add these can cause several challenges and limitations. This is because adding powers would require a really high level detection system which would add complexity to the existing collision system. Also each power would require animations and effects which also require more time and effort into the game and increase its complexity even more. I would also have to consider performance as I do not want the game to not be able to play on low to mid end computers, the aim is that everyone can play it.

- Potentially I could add simple powers which do not need high quality effects and animations so that it would not take a lot of my time and so all performance computers can run the game, Or these could be like an update where powers could come in the future of the games

- Having animated tiles and backgrounds for each level would keep the user very entertained as there would be constant animations real time. This would really upgrade the game and improve the way it looks. The problem is that I may not have enough time to implement this as this would be something I would at the end once all the essential things have been added into my game

- Randomly generated levels would make the user more engaged and would keep them playing for a longer time. It would make the game less boring since each level is different and it would be unpredictable. However this would be hard to implement because I would need some type of AI to create these levels and I may not have enough time to complete this

- Having a monster chase the character in the last level is something I would really like to add, however again I may not have time to implement this. I also would need to code all the different things required for this monster such as the controls and the collision system and how to make it chase the player throughout the level

- Custom characters are something that I thought about. It would make the user feel connected to the game since there is something that only he has within the game that no one else has. Having a two player mode would allow for friends to play together and to compete with each other. Having cinematic transitions between levels would also keep the user engaged. All of these features would make the game look better however I may not enough time to do this and I would also need to figure out how to code and implement all of these things

- Trying to add too many features may sound good but in the end if I do not have enough time to make sure all the features work well I will not have a well functioning game. Therefore I will need to focus on the main core parts of my game before trying to add anything which is not really essential

Success Criteria (essential)

Requirements	Justification
Minimum of 3 levels	Stakeholders have requested this, more levels will be added if there is time. This will give a good balance where the game isn't completed too quickly or where it will take too long
Main menu with important features	Through interviews with stakeholders I have concluded that the main menu has a help and leaderboard screen and a button to quit. This will inform the user of all the main things they can do.
Controls for character	Requested by stakeholders to have arrow keys to move left, right and to jump. These are the main controls needed to control the game.
Timer	Through analysis of stakeholder interviews found that this will be a good addition as it adds challenge to the game keeping the user engaged
Monsters and Spikes	Through analysis of stakeholder interviews found that this is needed in the game to make it fun as without it the game would be too easy and will leave the user bored.

Leaderboard Screen	Through analysis of stakeholder interviews found that this will start up competition though users who play the game so will be a good addition
Help Screen	Through analysis of stakeholder interviews found that this helps incase someone may not know the objective of the game or not know how to play it
Appeal to audience from children to teenagers	My game is intended for anyone but mainly children to teenagers, This is since my game will have different colours which a younger audience may find more appealing
Level 2 and 3 will be locked	I talked about how level 2 and 3 will not be accessible unless level 1 is completed and then level 2 will need to be completed for level 3 to be unlocked, this keeps the user from slipping earlier levels and moving straight to the last level
Background Music	This will be a good addition since it will add a vibe to the game
Let each level have a different Theme	This will be done through having each level with different backgrounds colours and different coloured tiles, this will keep the game entertaining since each level will be unique
5 lives for player	Through analysis of stakeholder interviews I found that this is a suitable amount of lives for the user, since it gives the user a few attempts but not too many
Have Bright and vibrant colours	Through analysis of stakeholder interviews I found that this will the look of the game and i will add this through the text and background colours
Level's Progressively get harder	This will make the game more challenging and fun so that the user does not get bored or feel like the game is too easy

Success Criteria (desirable)

Requirement	Justification

Monster Chase	I would like a final special level where there will be a monster which chases you down the level, however i do not know if i will implement this as i may not have enough time, this will keep the suspense of the game till the last level making the user keep playing the game to finish it
Transition Effects	This would make the transition between levels more engaging and less boring since it's better than just cutting to the next level
A way to regain lives	This would help the user because they might find the levels too hard and so this will give them more chances to try and beat the game

Design Requirements:

- Size Of Screen

  My screen will not fill the entire screen because I'm going for a specific retro look. From the interviews I've conducted with my stakeholders I've come to find they also do not like having screen which is too big and so therefore this size should be perfect for the user

- Graphics(background,scores,main page)

  For each level I will have a different background colour, I've talked about this with my stakeholders and they agree that this will enhance the playing experience and make the game look nicer.On the main page the user will have access to the help menu, leaderboard menu and can leave the game by pressing esc. Once they press enter they will start the game and they will spawn in. Score will increase once per second and to win you need to finish all three levels in the least amount of time.The tiles will change every level like the background colour, this also enhances the game experience by making it more fun and making the game look vibrant. I also talked about this with my stakeholders and they've agreed that it will make the game much better looking

- Object graphics (main player,baddies,items in game)

  My main player will be a red hooded knight throughout the entire game. This matches the games theme as its an escaping game. The enemies within my game consist of slime monsters and spikes. This gives variety as it's not only one monster and gives a challenge to the game. Linking back to the games i researched , Shovel Knight also has more than one enemy and through the interviews I've had with my stakeholders, they prefer more than one enemy as it makes the game more fun and challenging

- Text

There will be text displayed on the main menu screen displaying the name of the game as well as how to enter the help, leaderboard and playing state. To exit the game there will be text saying to press esc. In the help menu it will display how to play the game and the controls. The leaderboard will show the top 5-10 scores.

- Colours

For my game I wish to include vibrant colours which change throughout each level, this helps to make the game look good. I've spoken about this with my stakeholders within the interviews and I have gotten feedback from them saying that this is a good addition as it makes each level unique. Since my target audience varies from children to teenagers this makes sense as if i keep the background colour the same throughout the game it will make it boring. The tile colours will also change each level adding on to the fact that each level will feel unique

Input Requirements:

- How to enter game

To enter the game the user will just need to press enter on the main menu screen allowing for easy access. Levels will not be able to be chosen as I've discussed this with my stakeholders , I want the user to go through each level one by one. To Control the player the user will use the arrow keys allowing for an easy playing experience. Pressing the up arrow allows for the character to jump.

Processing Requirements:

When the enter key is pressed on the main menu screen the game "state" is switched into the playing "state" allowing the user to play the game. If the H key is pressed then the game state will be switched to the help screen state. If the L key is pressed then the game will switch to the leaderboard screen state. If the esc key is pressed then the game will close. When the character makes contact with the door at the end of each level the level selector selects the next level. If the character makes contact with the slime monsters or spikes then they automatically respawn in the beginning of the level, if all lives are lost the end menu screen is displayed and the state is switched to the end screen. For the character if the up

arrow is pressed then it will cycle through the jumping sprite sheet allowing for the jumping animation to occur. When Left or Right arrow keys are pressed the character will move left or right and show the animation by cycling through the spritesheet

Output Requirements:

Firstly, when my game is started the main menu is displayed with text showing the different options the user can take such as exiting the game, start playing the game, go to the help screen and go to the leaderboard. When the game is started the first level is displayed with the character, tiles, monsters, spikes , door and background colour. The score will also be displayed in the top left corner

Hardware And Software Requirements

Item	***Justification***
Keyboard	Allows the user to play the game and control the character
Mouse	Allows the user to start up the game and to navigate through the main menu into the help menu or into playing the game or to exit the game
Monitor	Allows the user to see the game
10Mb of hard drive space	The sprite images are 2kb each and the tile images also 1kb, the door image is 100kb so 10Mb should be enough
2Gb of RAM	This should be sufficient as the game isn't highly complex and so doesn't require much memory space
Intel/AMD dual core processor	An example would be Intel core 2 Duo or AMD Athlon 64 X2. A minimum of 1.6Ghz clock speed or higher. Pygame doesn't require heavy CPU processing unless complex physics is used
Python 3.x	Minimum of Python 3.0 needs to be installed as Pygame is a Python library and requires Python to run
Pygame Library	This is required as the game is made in pygame. This can be installed through the

	command “pip install pygame”
Windows 7 or higher	Windows 7 is required as it still supports Python 3.0 and Pygame, however for better performance windows 10 or higher is recommended
MacOS 10.9 or higher	Required as Python and Pygame are supported on MacOS 10.9 and newer
Any Recent Linux	Most recent linux distributions such as Ubuntu 16.04 or higher have Python already installed and then Pygame can be easily installed through pip

Computational Methods

Thinking Abstractly

- Abstraction is the process of removing unnecessary detail and only keeping the relevant parts. Abstraction is used to reduce the complexity and size of a program and to simplify it
- Movement of my character will be abstracted, the user will be able to move him left, right and up using the arrow keys, these are the main movements needed to move the character to complete the level
- My game will have a timer which increases and then outputs it as a score once the user loses all their lives. The internal details of how this works within the code isn’t shown to the user and in real life we don’t have a score so this is an abstraction
- My game also has monsters/enemies, these enemies cannot chase you and remain in the same position, they do not get tired and cannot speak, this is highly unlikely to happen in real life and so counts as abstraction
- When the character falls down from a tile placed at the top of the level to a tile placed at the bottom of the level, they will not take fall damage or get hurt , instead they will remain the same and the user can continue controlling them. This is an abstraction since in real life this wouldn't happen and so I’m only keeping the relevant details and not showing the user what would actually happen in reality such as getting hurt or losing health
- Also when the character walks into a tile at the sides of the level they just stop and then the user moves them the opposite way to continue playing. This is an abstraction since I am not showing the character actually hitting their head and body into the wall
- The transition from the current level to the next is abstracted since in my game you automatically get spawned into the next level. I do not show how this happens and this wouldn’t happen in reality and so therefore it counts as an abstraction

Thinking Ahead

- Thinking ahead is the process of brainstorming solutions for the outputs of certain inputs before creating the program, this allows you to think of potential solutions to problems you may come across later
- For my game the main inputs will include navigating through the different states such as the main menu, leaderboard screen and the help screen and then within the actual level the inputs would be moving the character using the arrow keys.
- In more detail, the first screen to be shown is the main menu, within the main menu the user will be able to move onto either the leaderboard screen or the help screen or exit the game by pressing esc, they will be able to move to the help screen by pressing the H key and the leaderboard screen by pressing L. Once in the game to control the character the arrow keys up, left and right will be used to move the character. Once all levels have been completed the game will ask the user for an input of their name to save their score to the leaderboard

Thinking Procedurally

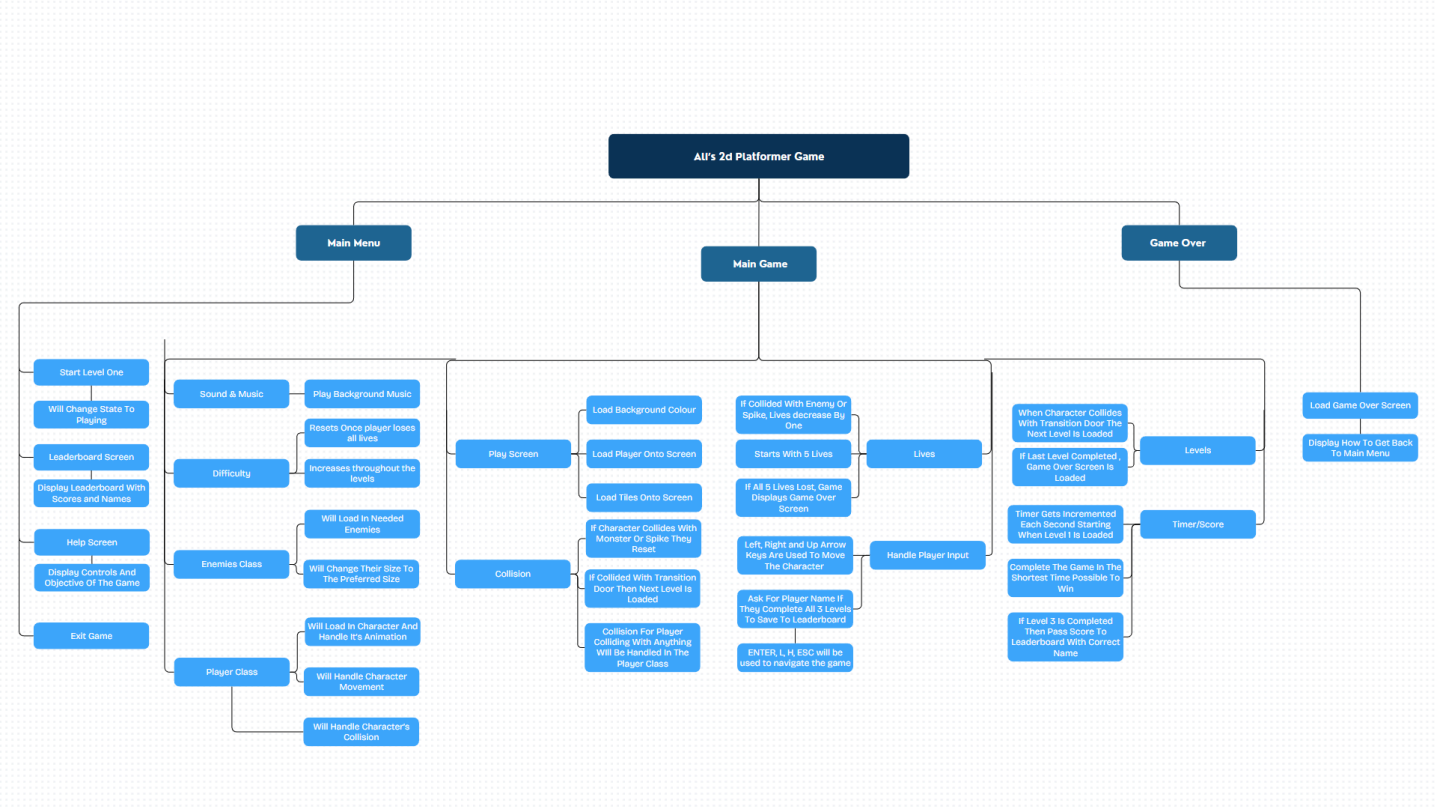
- Thinking Procedurally is the process of splitting a task into smaller tasks which are easier to carry out. These mini tasks all together build up to make the entire game.
- By doing this I will be able to use subroutines and classes within my games for different parts of the game. I will also need different states for the game , for eg , the main game state for playing, the man menu state for the main menu etc
- The way the different levels spawn in is the same, the only difference is the look of the level. Same for the character and background, these will use the same code for each level
- The enemies/monsters also spawn in using the same code, the only difference is where on the level they spawn in and when

Thinking Logically

- Thinking logically is the process of looking at the decision points within your code and seeing if it results in a different decision or enters a loop. These decisions affect the flow of the game and what happens.
- The main decision point in my game is the main menu and the different options the user has of selecting to enter the main game or entering the help screen or the leaderboard screen or to exit the game

Design

A structure diagram illustrating the problem decomposition.



Explanation of the different parts of the decomposition diagram

Function/Class	What It Does	Justification
Main Game	The Main Game loop will call the needed state/screen as the user navigates throughout the game	This is needed since if the user presses enter to play the first level the game will need to transition the playing state and not be stuck on the main menu
Main Menu	The Main Menu function will load the first image that the user will see, displaying the name of the game as well as all the other things that the user can do, such as moving to the help screen, leaderboard screen, exiting the game or loading the first level. This will be possible by pressing different keys such as ENTER to start playing level	This is needed since it is the first thing the user will see and will help the user navigate the game. It will also allow the user to enter level one to start playing the game which is essential as there would be no other possible way to start playing

	one, L for leaderboard screen, H for help screen and esc for exiting the game	
Leaderboard Screen	The leaderboard screen will display the top 5-10 scores achieved within the game with the name of the person who achieved it next to the score. It will use either an insertion sort or a bubble sort to keep the scores in order making it easier to see who has the lowest score (the lowest score will be number one as it indicates that they completed all three levels the fastest). It will also let the user to go back to the main menu by pressing ENTER	This is needed because it allows for different users to compete for the number one position making the game more fun keeping the users engaged. It is also needed to display the actual leaderboard
Help Screen	The Help Screen will display how to move the character within the levels using the arrow keys as well as the objective of the game which is to get through each level by going through the transition door	This is needed so that if the user is unsure of how to play the game and does not know what to do, they can check this screen which will inform them and answer a possible question they may have
Levels	This will check what needs to be loaded within the levels. It will probably scan a grid and detect within the screen if a tile, enemy, spike or the background needs to be loaded	This is needed as before actually displaying anything within the level, I need to check and know what needs to be displayed
Sound & Music	This will play background music while the user is playing the game to make sure the user does not get bored	This is needed as it helps in not making the user bored
Play Screen	This will actually display everything needed within the levels such as the character, tiles, enemies and background. It will need to check what the current level is and to display the correct level. It will need to show the score incrementing as well as the lives decrementing. It will check if the user has lost all	This is needed so that the user can see what the level actually contains so that they can play the game

	their lives and if they have then it will transition to the game over screen. It will need to check if the user has completed level three then pass the score to the leaderboard as well	
Game Over Screen	The Game Over screen will display that the game is over and the score that the user has achieved. If the user has completed all three levels then it will ask for the name of the user so that it can pass it to the leaderboard with their score to save it. If the user has not completed all three levels then it will not ask for any name input and just display that the game is over with the score achieved.This screen will then allow the user to go back to the main menu by pressing ENTER so that the user can see the leaderboard or try again if they want to	This is needed as it notifies the user that they have lost and can try again or if the user has completed all three levels then it notifies them that they have completed the game
Player	The player class will load the image of the character in which the user will use to move through the levels and size the character to the needed size. It will allow the character to move using the arrow keys and handle the animation of the character. It also will probably contain collision detection of the character to ensure the character cannot walk through anything and handle gravity allowing the character to jump	This is needed since the character is a core aspect of the game. The character allows the user to access each level and pass through them and allows the user to complete the game
Enemies	The Enemies class will load in the monster and change its size to what is needed. It will also help in making the monster move to the left and right increasing the difficulty of the game	

Basic Algorithms in pseudocode for the methods in the scripts.

Feature	Pseudocode	Justification
Menu Inputs	GameState=”Menu” Select GameState Case “Menu” If KeyHit (KEY_SPACE) Then GameState="Name" If KeyHit (KEY_H) Then GameState="Help" If KeyHit (KEY_L) Then GameState=”LeaderBoard” If KeyHit (KEY_q) Then Exit Case “Name” If KeyHit(KEY_ENTER) Then GameState = "Play” If KeyHit (KEY_ESCAPE) Then GameState = "Menu"" Case “Play” If KeyHit (KEY_ESCAPE) Then GameState = "Menu" Case “Help” If KeyHit (KEY_ESCAPE) Then GameState = "Menu" Case “LeaderBoard” If KeyHit (KEY_ESCAPE) Then GameState = "Menu" End Select	When you start the game you need to load the menu for the player to start the game. The first state will be the menu as I want it to be the first thing to be shown. Then the user will be able to move to the other states depending on what they want.
Collision Detection	Function Collision(xP,yP,widthP,heightP,xO,yO,widthO,heightO) If xP>= (xO+widthO) or (xP + widthP) <= xO Then Return False If yP>= (yO+heightO) or (yP + heightP) <= yO Then Return False Return True End Function	This will check collisions between the player and enemy. This will be needed so that if the player collides with an enemy, either they will die and lose a life or they will lose health. This is an important feature that will be needed

Bubble Sort	swap=True
	size=score.length()
	count = 0
	While (count <= (size-1) AND swap == True)
	swap = False
	FOR index= 0 to (size-(count + 1))
	if score[index] > score[index + 1] then
		This will sort the scores in order. This will be needed to create competition within the game and so that all the users can view who is first and what position they are in the leaderboard. It checks if the value1 is bigger than value2 and if it is then it sets swap to true and uses a temporary

	swap = True
	temp = score[index]
	score[index] = score[index + 1]
	score[index + 1] = temp
	endif
	next index
	count = count + 1
	endwhile
		variable to store value 2 while making the swap. It then goes to the next value and keeps repeating this until all values have been checked

Character Movement	Function Move
	move_x = 0
	Move_y = 0
	If User KeyHit (KEY_UP) Then
	Move_y = -10
	If User KeyHit (KEY_LEFT) Then
	move_x = -5
	If User Keyhit (KEY_RIGHT) Then
	move_x = 5
	End Function
		The purpose of this function is to handle the movement of the character. It allows the character to move either Left, Right or Up depending on the user's inputs. For example, if the Left arrow is pressed then the character's velocity will become -5 causing it to move to the left. It is needed because it allows the user to navigate through the levels and actually complete the game since without it then they wouldn't be able to do anything.

Grid Selection	Function Grid_Select
	row = 0
	For row = 0 to 20
	column = 0
	For column = 0 to 20
	If tile == 1 then
	Image = tile.png
	Image.x = position.x
	Image.y = position.y
	If tile == 2 then
	Image = enemy.png
	Image.x = position.x
	Image.y = position.y
	If tile == 3 then
	Image = door.png
	Image.x = position.x
	Image.y = position.y
	EndFor
	EndFor
	Endfunction
		The purpose of this function is to read through the rows and columns of a list and to assign specific images depending on the number contained in that position. For example if I want all four sides of my level to be covered in tiles so that the character cannot escape then I will write a 1 in the list for all four sides and so that when the code reads it it assigns a tile to those positions. This allows me to create multiple levels and to keep on using this same function for all the levels. It also allows for me to easily add anything I want in the future

Level Transition	Current_level = 1
	Level_data = level_data_1
	Level_number = 1
	Level_complete = False
	
	If level_complete == True then
	  Level_number += 1
	
	If Level_number == 2 then
	  Current_level = 2
	  level_data = levels_data_2
	
	If Level_number == 3 then
	  Current_level = 3
	  Level_data = level_data_3
	Endif
		The purpose of this is to handle which level is ot be displayed to the user so that they can play. The variable level_complete will be set to true whenever the user completes a level so that the game knows to go to the next level

Data structures to be used.

Variables

Method	Name	Data type	Purpose	Justification and Validation
Constant	Width	Integer	The purpose of the Width constant is to set how wide my game window will be in pixels. This will set how much of the game in width the user will be able to see.	- It is needed to be set as a constant as that makes sure the width of the screen remains the same at all times. Keeping it fixed also allows for more compatibility with display's and avoids problems with layouts - It should be set to positive values when the game is run to make sure user can see the game
Constant	Height	Integer	The purpose of the Height constant is to set the vertical dimension of the game window in pixels. It will set how much of the game the user will be able to see vertically	- It is needed to be set as a constant as that will ensure it stays the same at all times and will not accidentally get changed. It helps in keeping the intended screen layout so that I can add everything else within the game relative to it - During running of the game it should be set to positive

				values so the window actually shows up to the user
Constant	FPS	Integer	The purpose of the FPS (frames per second) constant is to choose the speed at which the game updates and renders frames. It keeps timing for everything constant	- It is needed as a constant to make sure that the game runs at a constant speed allowing for a smooth experience for users - During running of the game it should be set to a positive value and a daily chosen value to maintain smooth gameplay
Variable	State	String	The purpose of the ‘state’ variable is to track the current game state of the game whether that be the main menu, help screen, leaderboard screen or the play screen. It will allow for the game to switch between these states and then load what need to he loaded within these states	- Using a string for this variable makes it easy to read and understand, especially since each state name will describe roughly what gets displayed for it. This variable is important for making sure that only what needs to be shown to the user will get shown - For validation I could add checks to make sure that correct things are being passed to it, and if any incorrect values are given to it then it could revert back to the main menu or somewhere else
Variable	Window	Pygame Surface Object	The Purpose of the window variable is to set where everything within the game will be displayed. It will allow for user interaction to move between the states	- It is needed so that the user will be able to play the game since without it then nothing will be displayed and there would be no game to play

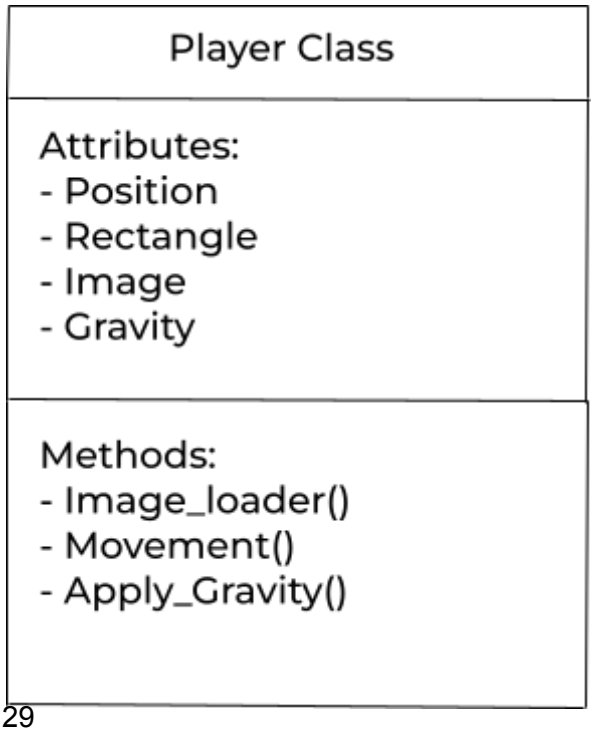
Variable	Level 1,2,3	List of list	The purpose of these list within lists is to represent the layouts of each level	- They are needed because they will allow me to create the levels and then modify them in any way I want. They will allow me to add things such as obstacles by setting a certain number to equal that obstacle or if I want a tile then a different number could then be used to represent the tile - For validation, they all need to be the same amount of rows and columns so that everything can be displayed neatly
Variable	Sprite_Speed	Integer	The purpose of sprite speed would be to control how fast the sprite animations of the character play when needed	- It is needed to maintain consistent visual effects to let the user know on the screen of the character is currently jumping, moving left, moving right or standing still
Variable	Lives	Integer	The purpose of the lives variable is to track the character lives during playing of the game	- This variable for the lives is needed because when the character might interact with enemies or obstacles and they collide with them I want the lives to decrease and the character to spawn back at its initial position. It will help in allowing the user to understand how the lives system works. It might also help in notifying when the user has lost all their lives and so that the game can then pull up the game over screen - It should always be kept as an appropriate positive integer to allow the user a good amount of tries
Variable	Score	Integer	The purpose of the score variable is to track the user's score while the character is still alive	- The purpose of this is to provide a way to measure how well the user did in playing the game and if they complete all the levels then they can save it to the leaderboard - Should be set to zero before game is started
Variable	Current_level		The purpose of this	- This is needed so that the

			variable is to store which level needs to be loaded depending on which level the user is currently playing	correct level will be shown to the user when they are playing. It will also help in transitioning between levels when the user completes a level - Always needs to store level one first before game is run and should appropriately change through the rest of the level when needed
Variable	Tile_size	Integer	The purpose of the tile size variable is to represent how big each tile within the level will be.	- Keeping the tile size the same throughout my game will be very important as I do not want different tiles to be different sizes. A consistent tile size will be important for collision detection later. - During running of the game it should be set to a positive value where it allows for the user to play the game easily
Variable	End_game	Integer	The purpose of the end game variable is to check whether or not the user has lost a life or not. If they have then it would be switched to something like -1 which will then cause the lives of the user in the game to decrement by one	- Using a variable to check whether or not the user has lost a life allows for easy understanding to control the game - For validation, It must be an integer with a defined value
Variable	Player	Instance of Player class	The purpose of the player variable is to represent the actual character the user will be moving during playing of the game	- It's needed as it will set the initial spawn position of the character. It will also track the player's location, movement and whoever they interact/collide with - Before game is run, it should set the spawn value ti somewhere within the level
Variable	Enemy_group		To manage all the enemies and the sprites used to display them	- It will be useful because it will allow me to customise the movement of the enemies if I wanted to and it will also be helpful in rendering them
Variable	World	Instance of the	Will be used to actual render the level layout	- It is important because it will be responsible to organising

		Levels Class	and tiles needed for each level while the user progresses throughout the game	and displaying the core elements of the game
Variable	Final_Level_Complete		Will indicate whether or not the user has completed the final level or not	- Will help in determining if the game should ask for the user's name or not because i haven't decided whether or not a user can enter their name for the leaderboard if they haven't finished all the levels
Variable	Name		Will store the name that the user entered in the game over screen	- This will allow for the users to see their name on the leaderboard screen and will create a sense of competitiveness for all who player the since everyone will be trying to get on the leaderboard

Class Diagrams

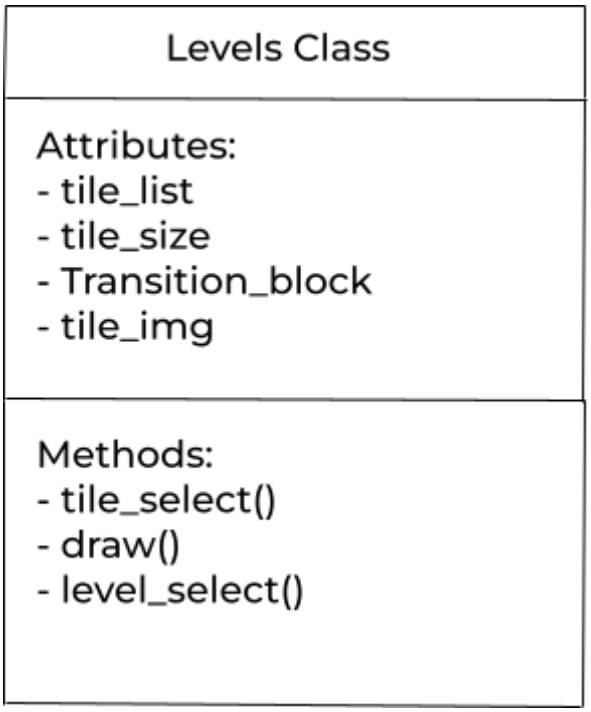
- First I will be making a rough plan for my player class which will be representing the character being moved on the screen. I will be adding the main attributes and methods I believe I will most likely use.



- The position attribute will set where the character will be spawning. This is needed because it will allow the user to see the character load in
- I will also need a rectangle of the character because I will be implementing collision detection and so this will help since it will act like a hitbox. This will allow the user to interact with other items in the game. The movement method will in this as it will handle the movement of the character and how they will interact with the other items within the game
- The Image attribute will load in the image I have chosen for my character and then the image_loader method will actually load it in. This will help the user in actually seeing what they are controlling to navigate throughout the levels

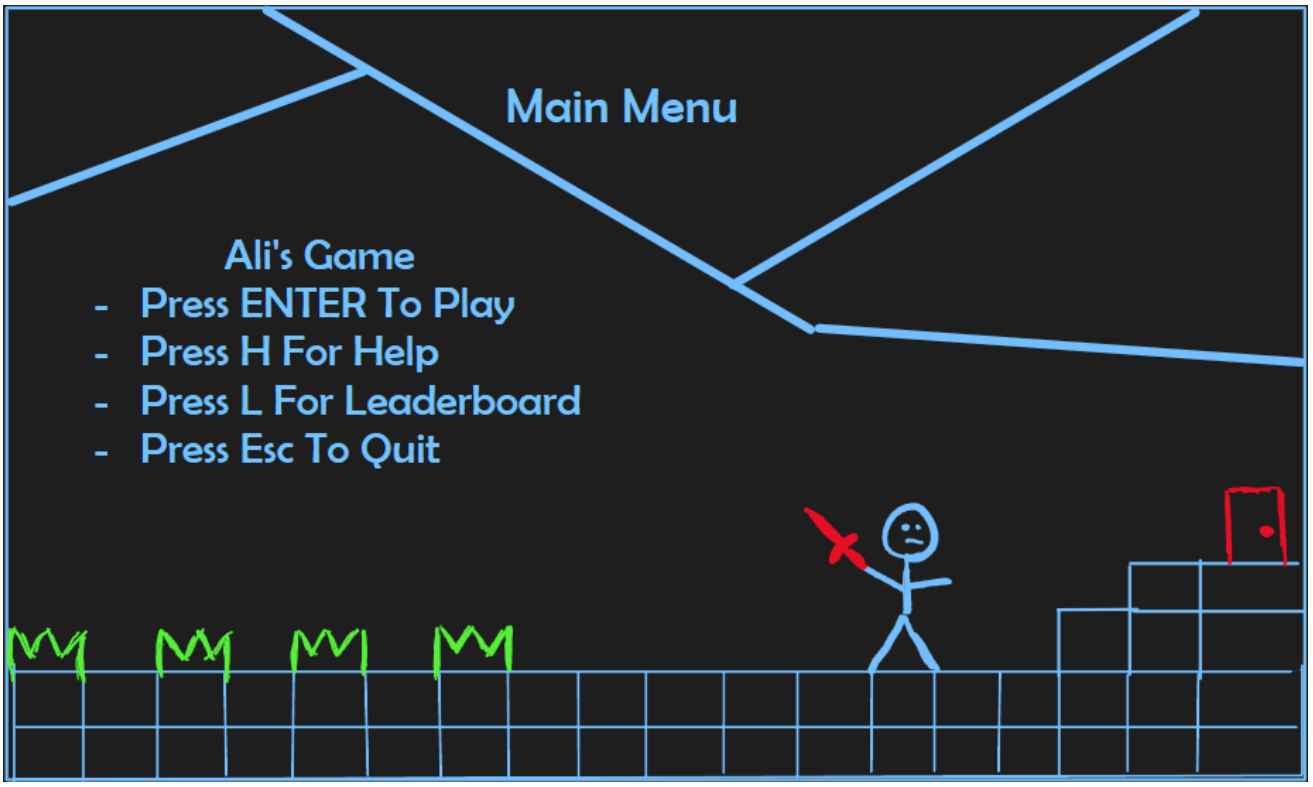
- The gravity attribute will be set with the Apply_Gravity method. These two will work together to make sure there is a somewhat realistic gravity effect for the player. This will give the game a much better feel and make it seem realistic

- Next is a levels class, this should handle everything to do with my levels such as what being displayed within the level

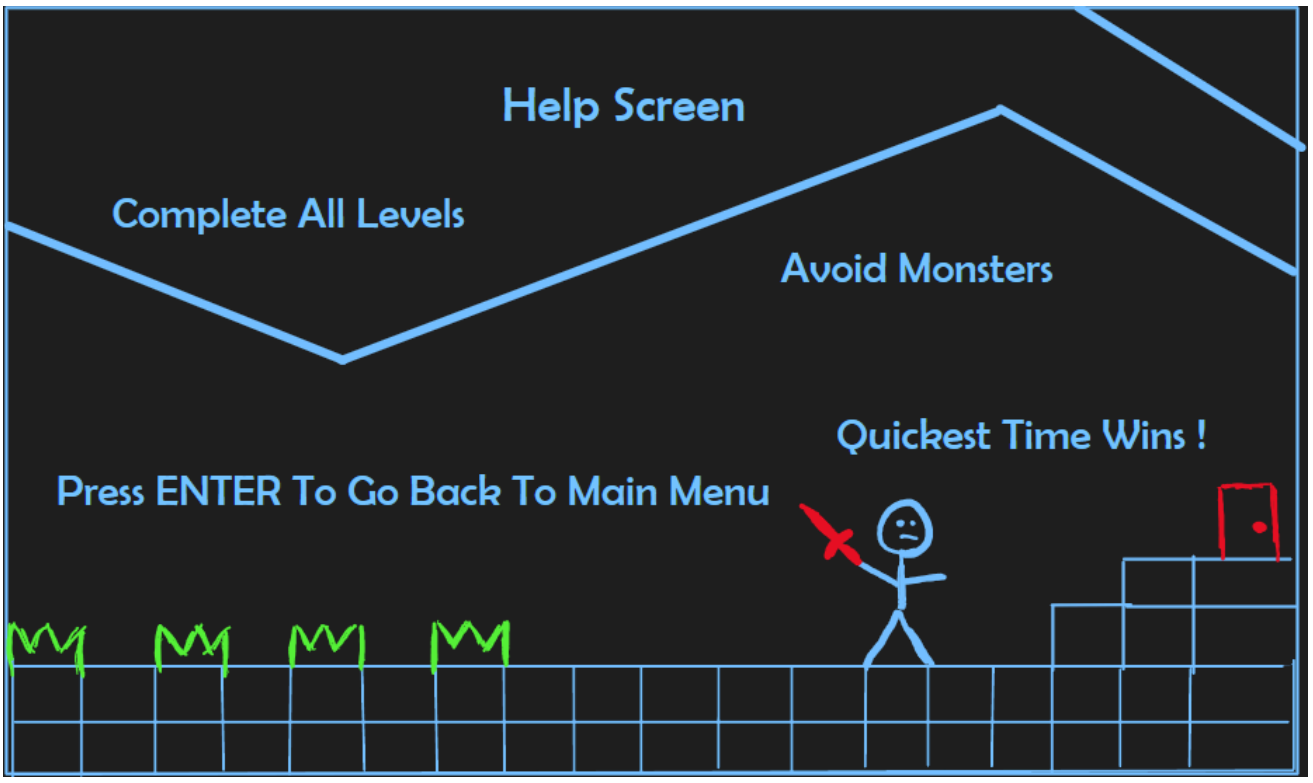


- The tile_list attribute would store the tiles and their positions, this will choose where each tile will be placed and create a pathway for the user to follow to complete the level. I will have to create this through trial and error or I might find a level creator which could create the levels for me.
- Tile_size will set the size of the tiles within the level and it will need to be set to a suitable size which will allow me to create a good level
- Transition_block will be the tile where when the character collides with it , it will transport him to the next level or to the game over screen if they have already finished all the levels
- Tile_img will be the actual image of the tile
- I am planning for the tile_select method to process tile_list to choose whether I want a tile or nothing in each position and this is how I will create the levels
- The draw() method will render all the level elements and output it to the screen

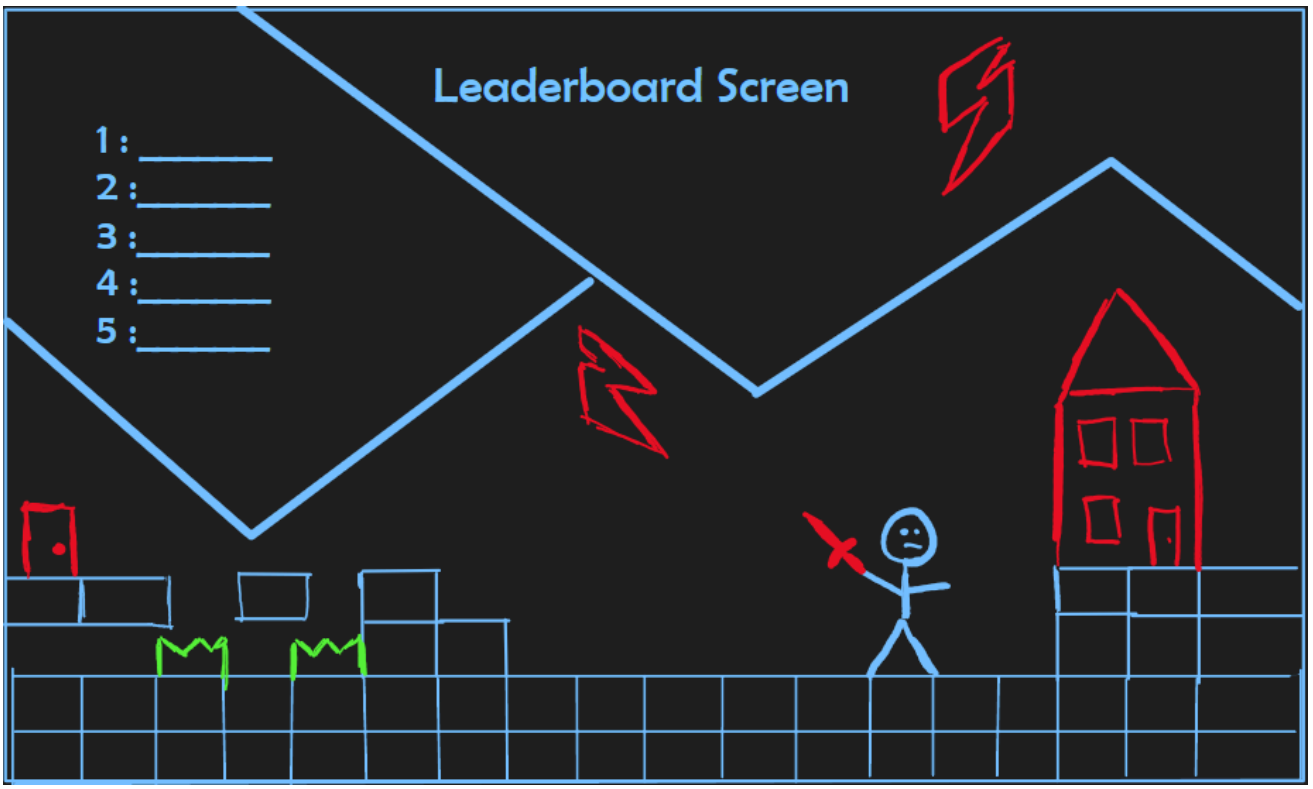
Screen Designs



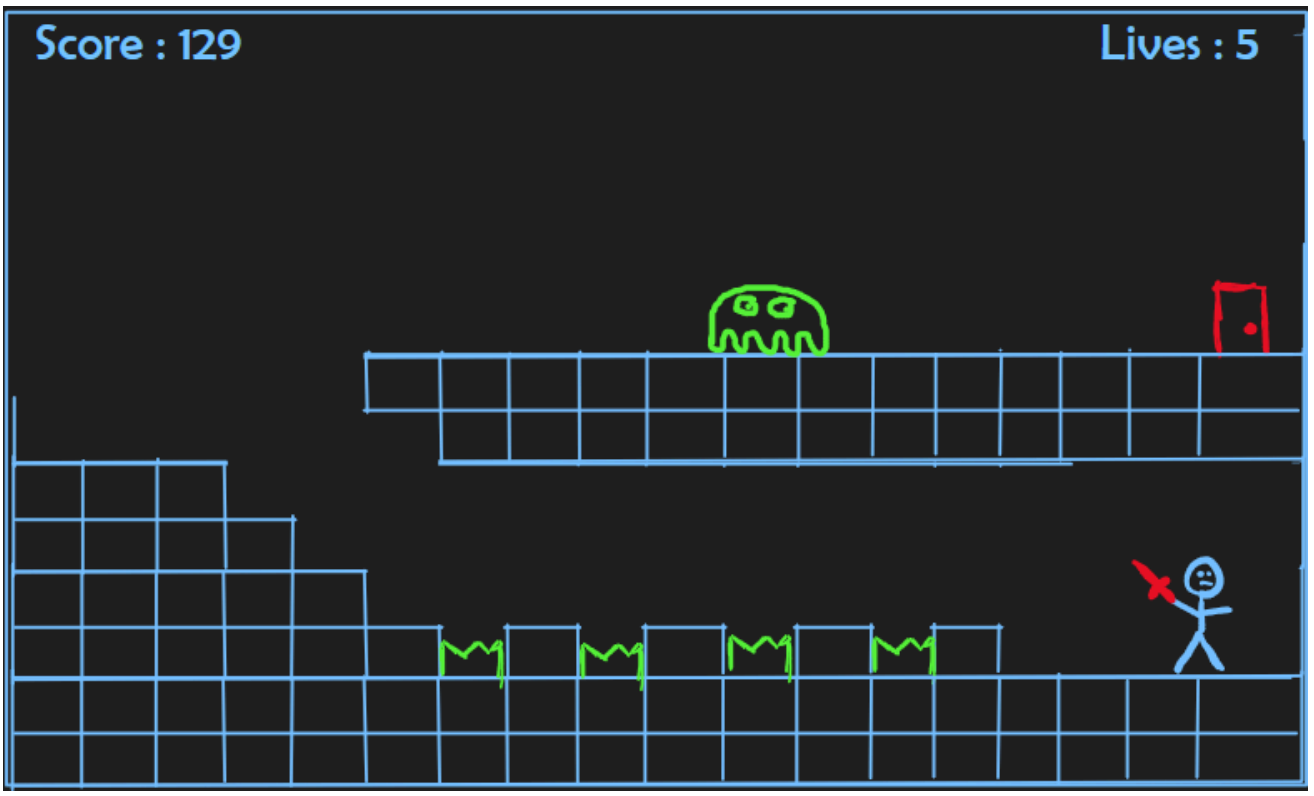
- The purpose of the main menu is that it will act as the starting point of the game. It will inform the user of the different screens which they can navigate to such as the help and leaderboard screen. It is important to have a well designed Main Menu screen as it sets the tone for the game. It should be visually appealing and easy to understand, that's why for my Main Menu screen I've put the different actions which the user can take with an appropriate size and a good position near the middle of the screen.
- I've drawn something similar as to what I'm hoping my game will look like as this will give the user an idea of what to expect. It's a character standing on tiles with a weapon ready to jump over the spikes within the level. This will get the user excited as it will make them want to play the game
- The random zigzag lines near the top of the screen help in making the screen look stylish, before I added them the screen looked fine but there was a lot of empty space and so it felt like something was missing, after adding these lines I feel like it makes the game look cooler and more visually appealing and that is what I need especially considering my target audience



- The purpose of the help screen is to inform the user on firstly, how to play the game and secondly, on any important objectives and mechanics which they should know of. This is because not everyone who plays the game will know what to do straight away. The Help screen should be easy to understand and read.
- Within my Help screen design I wanted to keep the character and tiles as that is a core part of what my game is about and I don't want the user to forget that. I've also kept the random zigzag lines and I've put a few possible instruction I may have in my actual game between the lines.I think this makes it more fun to look at as they aren't just one under the other, it forces the user to look around the entire screen and search for the instructions which is much more fun.



- The purpose of the leaderboard screen is to display to the user the top 5 scores with the names of the users who achieved those scores. This should motivate players and it adds a competitive feel to the game which in turn will increase the chances of users playing the game more than once
- I've kept but also changed the character and tile designs. Keeping them is important because it outlines to the user what my game is about but I've also changed it a little to add some variety to show the user the possible kind of look which the levels might have. I've added some cool lightning designs in red, this fills in empty space and makes the game look cooler which is important for my target audience
- The scores and displayed near the top left of the screen and are boxed up by the zigzag lines



- The purpose of each level is to display to the user what the level looks like. It will show where the character spawns in and where all the tiles are and the enemies. The user will have to figure out how to make it out of the level by reaching something such as a door which will teleport him to the next level
- As can be seen above I'm planning for the level to contain the score in the top left corner, this will always be visible and will let the user know the score they currently have, the lower the score the better since a lower score means they completed all the level faster. The top right corner contains the amount of lives the user has left, this can be useful for the user as they can think of whether to take risks or not depending on how many lives they have left.
- The spikes and monster will have to be avoided and jumped over to reach the door

Test data to be used during the development of the coded solution.

Test No	What I test	Reason	Expected Outcome

1	Loading the main menu	To check if the menu input algorithm is working and to see if the user will be able to load up the game and navigate through it	Game will load and will first show the main menu then the user will be able to select what they want to do next by looking at the different options displayed within the main menu
2	Loading the Help screen	To check if the menu input algorithm is working and to make sure the user can learn how to play the game	Game will first load with the main menu then by pressing a certain key the game will switch to the help screen which will inform the user of the objective of the game and the main things to avoid
3	Loading the Leaderbaord screen	To check if the menu input algorithm is working and to see if the user will be able to see the top 5 scores	Game will first load the main menu then the user will select to load the leaderboard screen through pressing a certain key then the leaderboard screen will be loaded displaying the top 5 scores with the names of the people who achieved those scores
4	Exiting the game	To check if the menu input algorithm is working and to see if the user will be able to exit the game	When game is first loaded, by pressing esc the user should be able to quit the game
5	Loading level 1	To check if the menu input algorithm is working and to see if the user will be able to load up the first level to play the game	When the main menu is loaded, by pressing a certain key the user can access the first level and start playing the game
6	Player Movement	To make sure that the user can move the character throughout the levels so that they can complete the game	When the user starts level 1 the character should move when the arrow keys are pressed
7	Level 1 display	To make sure that everything within the level is displayed correctly so that the user can	When level is displayed all the tiles and the character and the enemies should be displayed and spawn where they belong within the level

		play the game. This also makes sure that the grid selection algorithm is working fine	
8	Gravity	To make sure that the game feels realistic and the character within the game won't start flying	When the user starts up the first level and tries to jump by pressing the up arrow the character within the game should jump and then it should look like there is a force which pulls him down similar to gravity
9	Character Collision	To make sure that the user cannot just phase through tiles or enemies but instead has to jump over the tiles and avoid the enemies to finish the level	When the first level is displayed and the user starts playing, when the character reaches a tile they should stop and have to jump over the tile and when they reach an enemy they should have to jump over it
10	Level transitioning	To make sure that when the user completes a level then the game will transition them to the next	When the transition is required and the user completes a level then the game will display level 2 and spawn in the required objects where they belong
11	If the scores in the leaderboard screen are sorted	To make sure the sorting algorithm is working and so that the users will be able to see who has the best score	When the leaderboard screen requested by the user then the game will display the scores with the names of the users who achieved them in a sorted manner

Test data to be used post-development (Alpha Testing)

Test No	What I test	Reason	Expected Outcome
1	Starting the game	To see if the user will be able to load up the game to play it	Game will load and will first show the main menu

2	Loading level 1	To check if keys work in navigating through the game and allowing the user to start playing level 1	Level 1 will be loaded allowing the user to get started in playing the game
3	Loading leaderboard screen	To see if leaderboard screen will be shown to allows users to see if they beat others or not	Leaderboard screen will be loaded allowing the user to see scores of previous players Nothing should happen
4	Loading Help screen	To check if help screen will be loaded and if user will be informed on the important objectives of the game	Help screen will be loaded to inform user of important information of the game Nothing should
5	Exiting the game	To make sure the user can exit the game once they are finished with all three levels or when they get tired	Game will close
6	Going back to main menu from leaderboard screen	To check if the user will be able to navigate through the menus	The game should go back to the main menu after ENTER key is pressed
7	Going back to main menu from help screen	To check if the user will be able to navigate through the menus	The game should go back to the main menu after ENTER key is pressed
8	If game over screen loads when all lives are lost	To check if game knows when all lives have been lost	Game should transition to the game over screen once all 5 lives are lost
9	Going from game over screen to the main menu	To check if the user can move to the main menu once they lose all their lives	Game should transition to the main menu

Character Testing

Test No	What I test	Reason	Expected Outcome
10	If character can jump	To see if the character jumps as intended allowing the user to jump over obstacles and enemies when needed	Character will jump a height of just over 1 tile to allow them to go through the level
11	If character can jump over the slime monster	To see if its too easy or too hard to jump over the slime monster	Character should jump over the monster when monster gets too close
12	If character can move left and right	To check if user will be able to move character through the levels	Character should move left for a little then right
13	If character appears	If character does not appear then the user will not be able to play the game	Level 1 should load with the character in the bottom right of the level
14	If character falls through the tiles	If character falls through the tiles then there will be no character to move throughout the levels to complete the game	Level 1 should load and the character should be standing on the golden tile without falling through them
15	If character can jump onto other tiles	If the character cannot jump onto other tiles within the levels then the user will not be able to complete the game	Character should easily be able to jump onto another tile and stand on it
16	If the character dies and resets to spawn position when they collide with a slime monster	To make sure that the game is somewhat challenging and is not too easy and that the collision system is working	When the character collides with the slime monster the character should respawn at the bottom right of the level

17	If the character dies and resets to spawn position when they collide with the spike trap	To make sure the collision system between the spike and the character is working	When the character collides with the spike trap the character should respawn at the bottom right of the level
18	If the character stop moving up in their jump when they collide with a tile above them	To make sure the collision system between the character and the tiles working	Character stops jumping when the head of the character reaches the bottom of the tile
19	If the character stops moving when colliding with the tiles horizontally	To make sure the character cannot go off the screen and that the tiles stop them	Character should stop moving when colliding with the tiles

Level System Testing

Test No	What I test	Reason	Expected Outcome
20	If level one loads when needed	Level one should load when the user wants to play the game	Upon pressing the ENTER key the game should transition to level one
21	If level 2 loads once the character collides with the transition door in level 1	To allow the user to continue playing the game	Game should transition to level 2
22	If level 3 loads once the character collides with the transition door in level 2	To allow the user to continue playing the game	Game should transition to level 3
23	If level 1 load fine again once user completes all three levels and inputs their name for the leaderboard	To make sure game can be played again once completed	Game should load Level 1

Leaderboard/Score Testing

Test No	What I test	Reason	Expected Outcome
24	If timer works throughout all the levels or not	Timer is needed so that it can passed to the leaderboard with the name of the user	Game should transition t level 1 and timer should start and remain for all 3 levels
25	If game only asks for your name input and saves your score after completing level 3 and not level 1 or 2	The game shouldn't take name input or pass the score if all 3 levels have not been completed	Game should not ask for name input or save the scoreif level 3 has not been completed
26	If leaderboard displays correct information once level 3 has been completed	To check if the game will take in the correct user name input and display it with the correct score	Game should display the inputted name with the score achieved

Sounds Testing

Test No	What I test	Reason	Expected Outcome
27	Background Music	To check if there is background music while playing the game	User should be able to hear a sound of music while playing the game

Power Ups Testing

Test No	What I test	Reason	Expected Outcome
28	Monster chase	If levell 3 has a monster which chases you will you try to escape the level even faster	When the last level is reached , the user will be given a head start and then a monster will be spawned in which chases the user down throughout the level
29	Regain Lives	The level may be to	Throughout each level there will

		hard and so this is a way to give the user another chance	be a way for the user to gain a live
30	Transition Effects	If there are any effects while transitioning between levels	When a user completes a level, I can add a nice transition which opens up to the next level

Developing A Coded Solution

Creating General Structure For My Game 28/07/2024

The objective of this milestone is to set the basic structures for my game, this includes the different states such as play, level, game over and help state. It also includes setting the FPS and setting different keys to move through the states for user inputs.

```
1   # Importing Libraries
2   import pygame
3   import sys
4   import os
5   pygame.init()
6
7   # Setting basic structure for game
8   WIDTH, HEIGHT = 761, 762
9   FPS = 60
10  WHITE = (0, 100, 0)
11  BLACK = (0, 0, 0)
```

- First Here I Start by importing some libraries. Pygame is used to create games and sys is used to exit programs when the user quits the game. I then initialise all the pygame modules. I set some constants such as the width and height of my game, the fps and the background colour for now

```
15  # Setting the width, height and name of the game
16  window = pygame.display.set_mode((WIDTH, HEIGHT))
17  pygame.display.set_caption("Ali's platformer game")
```

- Here i create my game window using the dimensions set earlier and set the title of the game to my name

```
275    # Setting constants for each state of the game
276    START = "start"
277    PLAY = "play"
278    GAME_OVER = "game_over"
279    LEVEL_SCREEN = "level_screen"
280    LEADERBOARD = "leaderboard"
281    HELP = "help"
```

- Here I set some more constants, these are for the different states of my game, each state will represent a different screen which will display what is required. Implementing the different states like this makes it easier for me in the future since then I can create a function for each state and display what I want

```
#Function to draw text for each state
def draw_text(text, size, color, x, y): #Creating a function to draw text onto screen
    font = pygame.font.SysFont("comicsans", size) #Assigning the font for the text
    label = font.render(text, True, color) #Render as image
    window.blit(label, (x, y)) #Blit image to screen
```

- Now to draw the text I need to create a function for it. The arguments I'm passing in are text which is what I want to write, size which is how big i want my text to be and colour and the x and y coordinates of where the text will be placed. I then render the font as an image and then blit (draw) the text image onto the screen

```
30  def main_menu():
31      run = True
32      while run:
33          window.fill(WHITE)
34          draw_text("Start Screen - Press Enter to Play", 40, BLACK, WIDTH // 4, HEIGHT // 2)
35          draw_text("Press H for Help, Esc to Quit", 30, BLACK, WIDTH // 4, HEIGHT // 2 + 50)
36          pygame.display.update()
```

- I then create a function for my main menu and set run to true so that it becomes a loop which will keep running until the user changes the state.I fill the screen with white for now and draw some text displaying this as the starting screen. Lastly i update the screen

```
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    run = False
                if event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_RETURN:
                        return PLAY
                    elif event.key == pygame.K_h:
                        return HELP
                    elif event.key == pygame.K_ESCAPE:
                        pygame.quit()
                        sys.exit()
```

- I then add in this loop which will check for user inputs and events and it will allow the user to start the game by pressing enter, go to a help screen by pressing h and exit the game by pressing esc

```
def play_screen():
    run = True
    while run:
        window.fill(WHITE)
        draw_text("Play Screen - Press L for Level Screen, G for Game Over", 40, BLACK, WIDTH // 8, HEIGHT // 2)
        pygame.display.update()
```

- This function will handle my playing state. Like the main menu it fills the screen with a white background and displays some text and then updates the screen for now

```
# For loop to allow user to navigate through the game
for event in pygame.event.get(): # Checks for events
    if event.type == pygame.QUIT: # Checks if user wants to quit
        run = False # Closes application
    if event.type == pygame.KEYDOWN: # Checks if a key is pressed
        if event.key == pygame.K_RETURN: #Checks if Enter key is pressed
            return PLAY # Outputs playing state
        elif event.key == pygame.K_h:  # Checks if H key is pressed
            return HELP # Outputs help state
        elif event.key == pygame.K_l:  #Checks if L key is pressed
            return LEADERBOARD # #Outputs Leaderboard state
        elif event.key == pygame.K_ESCAPE:  # Checks if esc key is pressed
            pygame.quit() # Closes application
            sys.exit()
```

- This code here also handles user inputs, if the user presses L it goes to the level screen and if they press G it goes to the game over screen

```
# Creating a function for the game over state
def game_over_screen():
    run = True # Sets the state to true
    while run: # While run is true, the following will happen...
        window.fill(WHITE) # Fills the screen with white colour

        draw_text("Game Over Screen - Press Enter to go to Start Screen", 40, BLACK, WIDTH // 8, HEIGHT // 2) # Displays the text
        pygame.display.update() # updates the screen

        for event in pygame.event.get(): # checks for events
            if event.type == pygame.QUIT:
                run = False # If user wants to quit , it will stop running
            if event.type == pygame.KEYDOWN: # checks if any keys are pressed
                if event.key == pygame.K_RETURN: # checks if enter key is pressed
                    return START # outputs the main menu
                elif event.key == pygame.K_ESCAPE: # if the escape key is pressed
                    pygame.quit() # exits the game
                    return START
```

- Again i create another function for the game over state filling it in with white for now and displaying text. If the user presses enter it goes back to the main menu and esc to quit

```
83  def level_screen():
84      run = True
85      while run:
86          window.fill(WHITE)
87          draw_text("Level Screen - Press Enter to Play", 40, BLACK, WIDTH // 4, HEIGHT // 2)
88          pygame.display.update()
89
90          for event in pygame.event.get():
91              if event.type == pygame.QUIT:
92                  run = False
93              if event.type == pygame.KEYDOWN:
94                  if event.key == pygame.K_RETURN:
95                      return PLAY
```

- I also create a function to handle my level state, fill it white white and display some text.If the user presses enter it will go to the play screen

```
405  # Function to create leaderboard screen
406  def leaderboard_screen():
407      run = True
408      while run: # While run is set to true..
409          window.fill(WHITE) #Fills the screen with white colour
410          draw_text("Leaderboard Screen - Press Enter to go to Start Screen", 40, BLACK, WIDTH // 8, HEIGHT // 2) # draws text
411          pygame.display.update() # updates the screen
412
413          for event in pygame.event.get(): # checks for event
414              if event.type == pygame.QUIT:
415                  run = False # Closes the state
416              if event.type == pygame.KEYDOWN: # checks if any key is pressed
417                  if event.key == pygame.K_RETURN:  # Checks if enter key is pressed
418                      return START # returns to main menu
419
```

- I also create a function to handle my leaderboard state, this will output the top five scores. For now I draw some text to know that it is my leaderboard screen and create an event handler

```
# creating a function for the help state
def help_screen():
    run = True
    while run: # while run is set to true..
        window.fill(WHITE) # screen gets filled with white colour
        draw_text("Help Screen - Press Enter to go to Start Screen", 40, BLACK, WIDTH // 8, HEIGHT // 2) # draw the text
        pygame.display.update() # updates the screen

        for event in pygame.event.get(): # checks for any events happening
            if event.type == pygame.QUIT:
                run = False # Closes the state
            if event.type == pygame.KEYDOWN: # checks if any key is pressed
                if event.key == pygame.K_RETURN:  # checks if any key is pressed
                    return START # returns to main menu
```

- I also create a function to handle my help screen, this is intended to state the objective of the game and the controls in case the user does not know how to play. I also draw some text for this screen for now and create an event handler for it

```
# main game function
def main():
    clock = pygame.time.Clock() # helps control the game's FPS
    state = START  # program starts in the start state
```

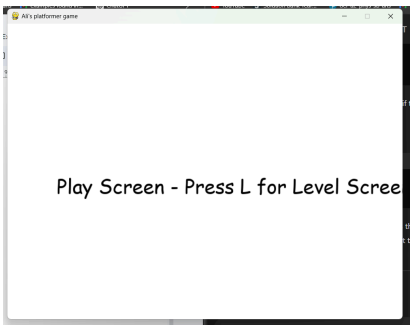
- Now this function is the main game loop. I set clock to control the game's FPS and make the game begin in the start state

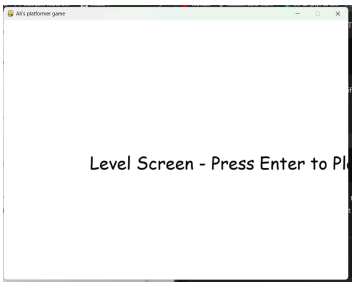
```
while True:
    clock.tick(FPS) #sets the the FPS at 60
# The following code called the needed state
    if state == START:
        state = main_menu()
    elif state == PLAY:
        state, score = play_screen()
    elif state == GAME_OVER:
        state = game_over_screen()
    elif state == LEVEL_SCREEN:
        state = leaderboard_screen()
    elif state == LEADERBOARD:
        state = leaderboard_screen()
    elif state == HELP:
        state = help_screen()
```

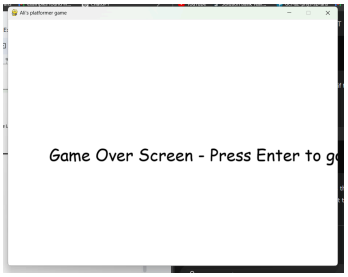
- The second line makes the game loop at 60 fps as I set it to 60 in the beginning of the code. To manage the states I use if statements to call whatever state is currently needed

```
        # closes application if user wants to quit
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()

# makes sure the games only runs if the game is called directly
if __name__ == "__main__":
    main()
```

- Here I check if the user wants to close the window the game can actually close and make sure that the game runs only when the code is called directly to not cause problems







- Here is the game in action , as you can see when i press enter from the start screen it changes to the play screen and from the play screen i can change to the level screen and game over screen, this shows i have achieved the objective for this milestone and completed it successfully

Creating The Player - 30/07/2024

- The aim of this milestone is to create my player, to do this i will create a class for the player and set the values for it. I then will blit the image to the window and create an instance of the player then update the playing state so that it can draw the player

```
24 class Player():
25     def __init__(self, x, y):
26         img = pygame.image.load('hoodguy1.png')
27         self.image = pygame.transform.scale(img, (40, 80))
28         self.rect = self.image.get_rect
29         self.rect.x = x
30         self.rect.y = y
```

- Here i create the class for my player then initialise it and take in the x and y coordinates as arguments, I then load my image to the variable called img. Then the next line helps me make the character twice as tall as he is wide. I then create a rectangle for the image and set the coordinates for the rectangle so that I can use it in the future for collisions

```
32     def update(self):
33
34         window.blit(self.image, self.rect)
```

- I also add this function in the class just for now it only helps blit the image at the position of the rectangles

```
player = Player(650, HEIGHT - 130) # creating an instance of my player and setting his spawn position
```

- I then create an instance of this player and make his position starting at the bottom block

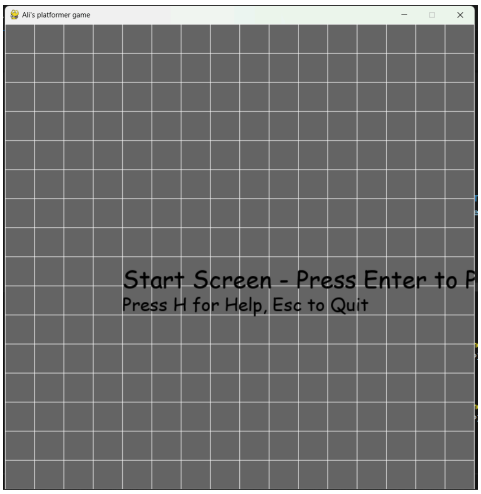
```
136         player.update()
```

- Then in the playing state i draw the character

- Here you can see the character is displayed meaning the code works and so the milestone has been achieved

Creating Level 1 - 05/08/2024

- The objective for this milestone is to create the first level with all the tiles. To do this I will create a list where different numbers will represent the tiles and enemies and whatever I want to add in the future. I will also create a class for it which will go through each row and decide what to display in the level.




```
9    WIDTH, HEIGHT = 785, 800
```

```
19  def draw_grid():
20      for line in range(0, 20):
21          pygame.draw.line(window, (255, 255, 255), (0, line * tile_size), (WIDTH, line * tile_size))
22          pygame.draw.line(window, (255, 255, 255), (line * tile_size, 0), (line * tile_size, HEIGHT))
23
```

- To create my level I will be using these grid lines, these will help since it allows me to see where I can place the tiles for each level. It also allows me to imagine how the level will look like and help me design it

```
49    game_data = [
50    [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
51    [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
52    [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
53    [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
54    [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
55    [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
56    [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
57    [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
58    [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
59    [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
60    [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
61    [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
62    [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
63    [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
64    [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
65    [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,],
66    ]
```

- This list here presents the grid layout of the level, a 1 represents the tile i am placing and a zero represents nothing, This allows me to easily create levels and I can also easily modify them however I want

```
# Function for levels
class Game_data():
    def __init__(self, data): # Constructor taking in self and data as arguments
```

- To actually read the list and extract the useful information I need from it I will create a class, I start it with the constructor which will take in the self and data argument (the data is the list i created).

```
img = pygame.transform.scale(gold_img, (tile_size, tile_size))
```

- I first start by loading in the image that I will be using to represent the number 1 from the list, what I want this part of the code to basically do is run through the rows one by one and within each row check each tile and show what it needs to represent, like if it's a 1 show the gold block or if it's a zero show nothing

```
class Game_data():
    def __init__(self, data): # Constructor taking in self and data as arguments
        self.tile_list = [] # creating a list
        gold_img = pygame.image.load('gold.png') # loading in the gold tile image
        silver_img = pygame.image.load('silver.png') # loading in the silver tile image

        row_count = 0 # start from first row and continues
        for row in data:
            col_count = 0 # starts from first column and continues
            for tile in row:
                if tile == 1: # This checks if a 1 is present
                    img = pygame.transform.scale(gold_img, (tile_size, tile_size)) # transforms image to preferred size
                    img_rect = img.get_rect() # creates rectangle for image
                    img_rect.x = col_count * tile_size
                    img_rect.y = row_count * tile_size
                    tile = (img, img_rect)
                    self.tile_list.append(tile) # adds to list
                    col_count += 1 # increments by one to next column
            row_count += 1 # increments by one to next row
```

```
13    tile_size = 49
```

- To achieve this I create a for loop which checks each row line , first it checks the row, then within the row checks each tile and if it's a 1 then it will show the gold block image. I have also used the transform function within pygame to scale up the image to the tile size i set earlier
- I then create a rectangle object (just takes the size of the image and creates a rectangle from it) as this will be useful in the future for collision and much more
- I set the column count and row count to zero in the beginning so that when the list is read it can move on the the next column and row and continue reading the entire list
- I create a tile list in the beginning so as the code goes through the game_data list the values will be added to it
- So essentially this code will go through game data list and take all the useful things such as the number 1 and ignore all the zero's and then get the coordinates for the rectangle for their rectangle and will store it in the new list

```
world = Game_data(game_data) # creating an instance by passing in the level layout
```

- I then create an instance of it by creating the world , calling the world class and then giving it the game_data list

```
# method which will will iterate through the tile list i created
def draw(self):
    for tile in self.tile_list:
        window.blit(tile[0], tile[1])
```

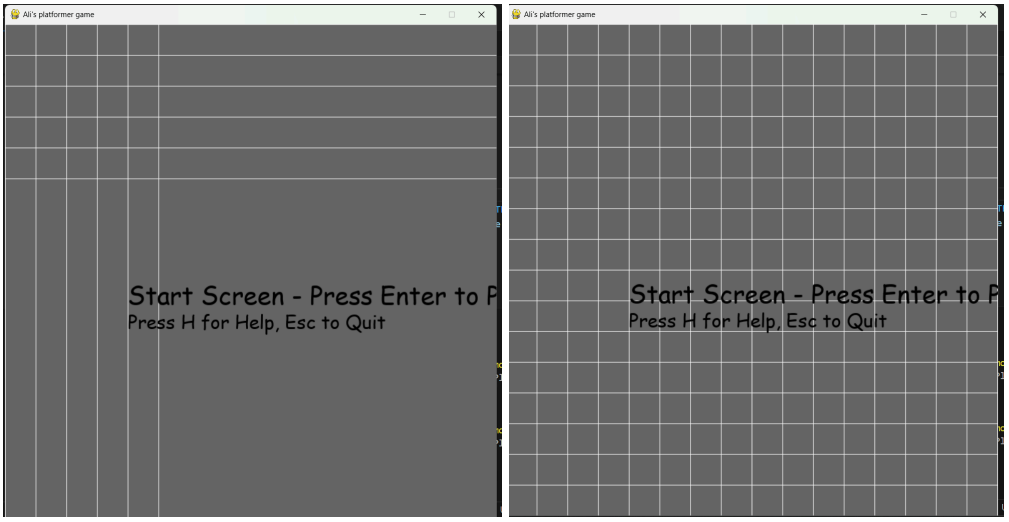
- Within the game_data class i then create a new method which will iterate through the tile list i created

```
world.draw() # draws the tiles
```

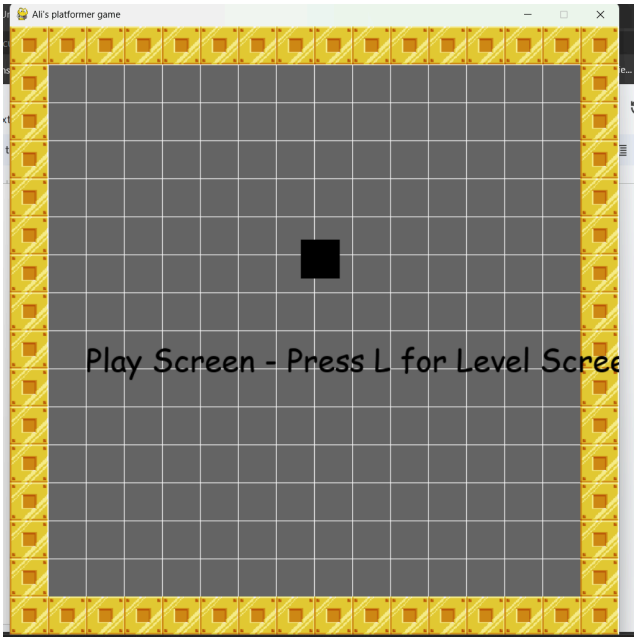
- This line here actually draws the tiles for me

Problem 1, Fixing Grid lines - 05/08/2024

```
20    for line in range(0, 6):
```

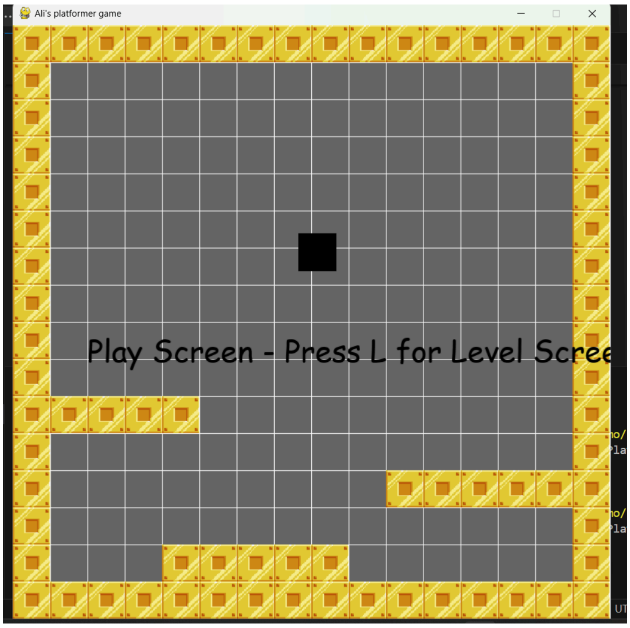


- When i first wanted to check if the grid lines where correct or not i saw this, there was grid lines but not enough so i went through my code to the function for it and turns out I was only writing for 6 lines so i then changed that to 20 which made my grid lines fill the entire screen



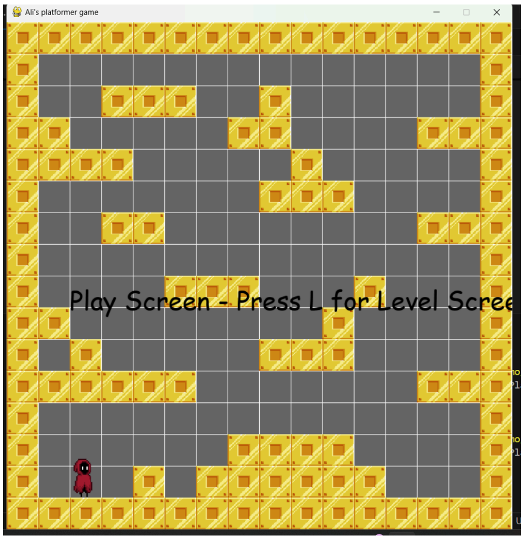
Once i press play I also can see the gold blocks filling the grid boxes just like the list i created, i can customise the blocks straight from changing the list

```
game_data = [
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,1],
[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1],
[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,0,0,0,1,1,1,1,1,0,0,0,0,0,0,1],
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,],
]
```



- As you can see near the bottom i have changed a few zero's to one's and you can see the change take place

```
game_data = [
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,0,0,1,1,1,0,0,1,0,0,0,0,0,0,1],
[1,1,0,0,0,0,0,1,1,0,0,0,0,1,1,1],
[1,1,1,1,0,0,0,0,0,1,0,0,0,0,0,1],
[1,0,0,0,0,0,0,0,1,1,1,0,0,0,0,1],
[1,0,0,1,1,0,0,0,0,0,0,0,0,1,1,1],
[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,0,0,0,0,1,1,1,0,0,0,1,0,0,0,1],
[1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,1],
[1,0,1,0,0,0,0,0,1,1,1,0,0,0,0,1],
[1,1,1,1,1,1,0,0,0,0,0,0,0,1,1,1],
[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,0,0,0,0,0,0,1,1,1,1,0,0,0,0,1],
[1,0,0,0,1,0,1,1,1,1,1,1,0,0,0,1],
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,],
]
```



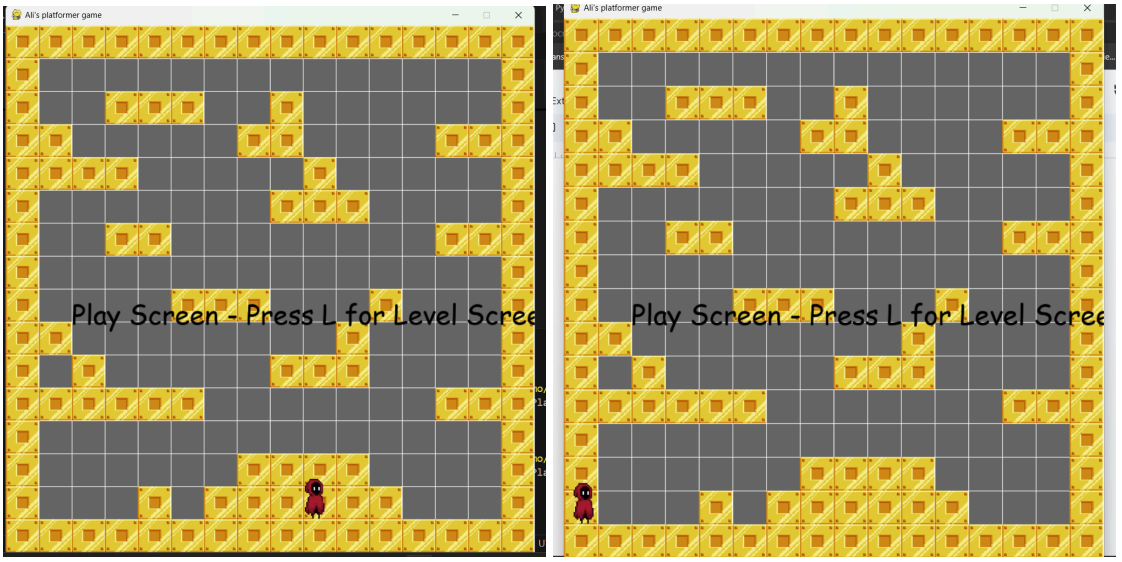
- After some trial and error I have got a base level which I can change in the future if wanted to
- Reviewing the milestone, I have achieved it as you can see, my level 1 is being displayed fine and the game is ready to move on

Adding Basic Controls For The Player - 07/08/2024

- The objective of this milestone is to create the controls for the character so that I will be able to move left,right,up and down. To do this I will add the controls in the update method of the player class

```
32      def update(self):
33          checkx = 0
34          checky = 0
35          key = pygame.key.get_pressed()
36          if key[pygame.K_LEFT]:
37              checkx -= 5
38          if key[pygame.K_RIGHT]:
39              checkx += 5
40
41
42          self.rect.x += checkx
43          self.rect.y += checky
44
```

- In the player class, the update method is where I will be adding in the controls. I want the collision function to be able to somewhat check if there is a collision before the player actually reaches a collision to see whether the character can move there or not so that if he can't he will stop. This allows the collision function to be checked before the player actually reaches a collision. So instead of moving the player's rectangle straight after the key has been pressed i created two variables checkx and check y which are changed after the key has been pressed.
- I then update the rectangle at the end with new position of the character



- As you can see here i can move the character now left and right

Adding Basic Controls For The Player PT 2 - 10/08/2024

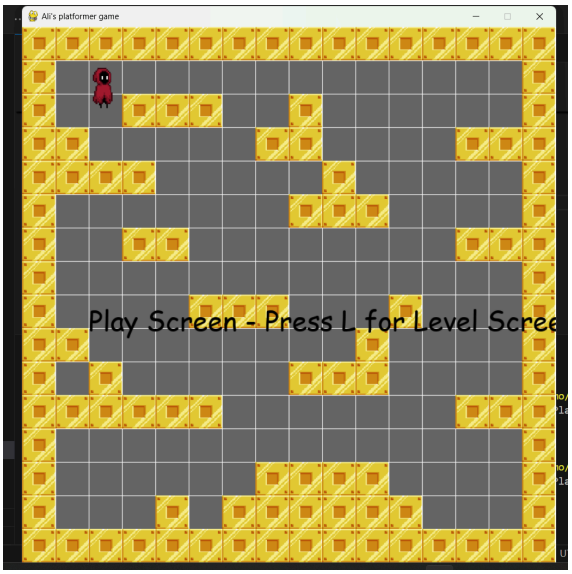
```
31                  self.vel_y = 0
```

```
key = pygame.key.get_pressed()
if key[pygame.K_UP]:
```

- I want the jumping with gravity to be somewhat realistic so i'm not gonna use the same way of checkx and checky to move left and right. I set the y velocity to zero then when he presses up he will move up with 15 pixels

```
checky += self.vel_y
```

- I then set the velocity he will be moving up with to the check y variable so that it increases and the player moves up



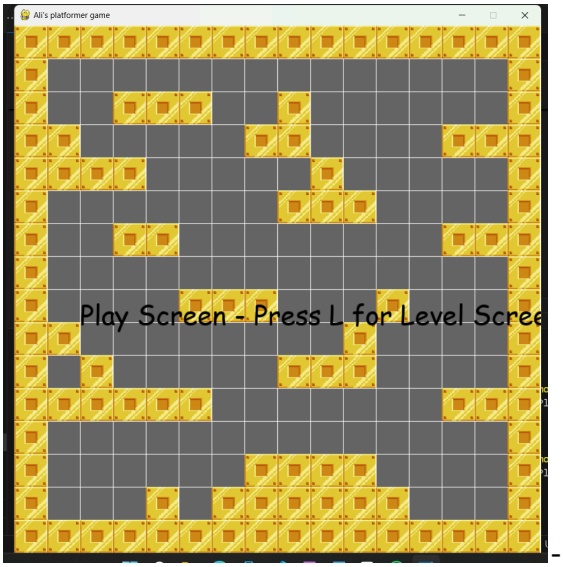
- As you can see here the character just moves up infinitely when i press up and goes off the screen as currently there is no gravity
- Reviewing this milestone, I have completed the objective as I have a character which I can move in four directions

Adding Gravity For The Player - 15/08/2024

- The objective of this milestone to add gravity into my game so that when the user presses jump on the character, the character will come back down onto the tile making the game more realistic

```
self.vel_y += 1 # slowly increases velocity to make gravity realistic

# limiting the speed so player does not fall too fast
if self.vel_y > 10:
    self.vel_y = 10
```

- All i need to do is when i jump the y velocity is -15 pixels making the character go up i add 1 to it to make it go back down but i don't want it to infinitely keep going up so once it reaches higher than 10 it gets set on 10



- Now my player falls down straight away as there is no collision but if i hold the up key the character will come flying back up

Problem 2, Infinite Jump - 16/08/2024

- A problem which i face now is that when i hold the up key my player keeps going up, it shouldn't do that as i don't want my character to just keep flying, i want him to only go up a little then i want gravity to take him back down the ground

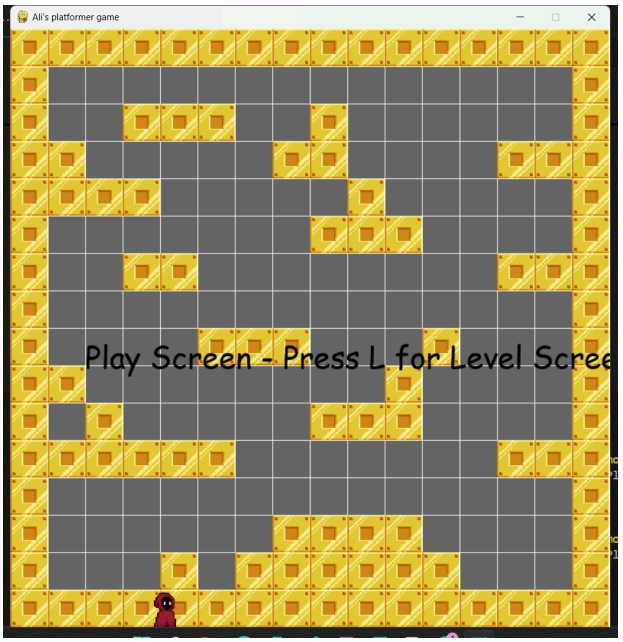
```
32          self.checkjump = False
```

```
key = pygame.key.get_pressed() # checks all the keys

# checks if the UP arrow is being pressed
if key[pygame.K_UP] and self.checkjump == False:
    self.vel_y = -15 # velocity for jumping
    self.checkjump = True # helps prevent spamming jump

# resets if jump is not being pressed
if key[pygame.K_UP] == False:
    self.checkjump = False
```

- To overcome the problem I mentioned earlier about the fact that if I hold up the player will just keep flying up, I created a trigger. I created a variable named checkjump and it stays false until i press jump which then turns it into true and once i release the up key it goes back to false, so that the user cannot hold up and keep flying

```
# checks if player has fallenn below screen
if self.rect.bottom > HEIGHT:
    self.rect.bottom = HEIGHT
    checky = 0
```

- I also added this for now so that the character does not keep falling



- Now i can see my character and i can jump however if i spam the up arrow key my character will keep flying up but i can't fix that until i get my collision sorted so it's just gonna have to wait

- Reviewing the milestone, I can say that i have achieved the desired output as now my character is affected by gravity and is always being pulled down when there is no tile under them

Adding Sprite Animation For Player - 16/08/2024

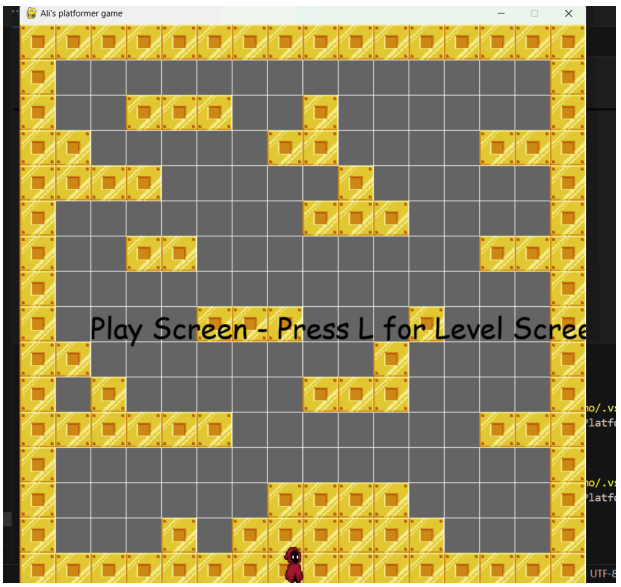
- The objective of this milestone is for my character to have animation when moving left or right, this is so that the game looks visually more appealing and engages the user. To achieve this I will be using a set of images which I have numbered and cycle through them whenever the user moves the character left or right

```
25      def __init__(self, x, y):
26          self.images_right = []
27          self.index = 0
28          self.counter = 0
29          for num in range (1,6):
30              img_right = pygame.image.load(f'hoodguy{num}.png')
31              img_right = pygame.transform.scale(img_right, (40, 65))
32              self.images_right.append(img_right)
33          self.image = self.images_right[self.index]
```

- Here what i did so far is create a list and an index that starts from zero so that i can track which image I want from the list. I also create a counter variable and set it to zero so that I can control the speed at which the animation runs.
- To load in the images I use a for loop and f strings to cycle through each of the images as I've numbered them 1 to 5 and I scale the images to my preferred size.
- I then append the images to the actual list so that it can store them and the next line makes sure it starts from the very first image

```
55          #handling the animation
56          self.index += 1
57          if self.index >= len(self.images_right):
58              self.index = 0
59          self.image = self.images_right[self.index]
```

- Now to handle the actual animation what i've tried to do is increase the index by 1 so that the next image is shown and so i increase the self.image variable by the index. I also make sure that if the index finishes it restarts again from the first picture by setting it to zero

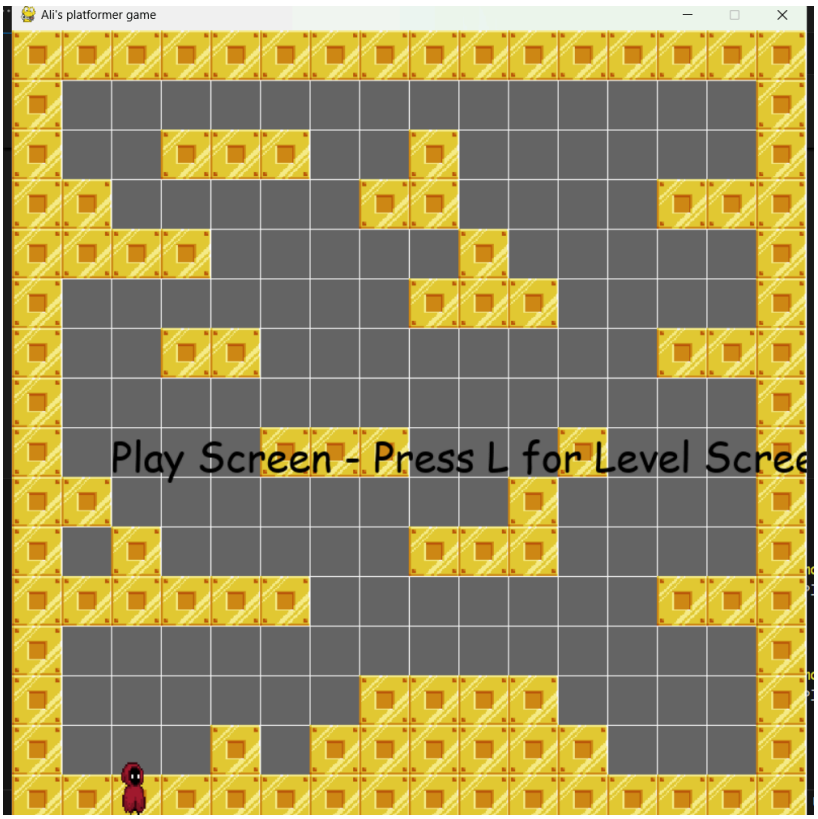


- Now it may not show in this picture above but what's happening now is that its iterating through the images at a very fast speed and iterating through them as fast as it can

```
44          sprite_handler = 20
```

```
self.counter += 1
if self.counter > sprite_handler:
    self.counter = 0
```

- Now to handle the speed and slow it down I create a variable called sprite handler and set it to 20. I use the counter variable I created earlier that i set to zero and increment it by 1 and so that when it reaches 20 it resets back to zero, now this should slow it down by a lot



- Now again you may not be able to see it but the animation is much slower and i can see the animation working

Problem 3, Sprite Iteration - 19/08/2024

- However the problem now is that I don't want it to keep cycling when I'm not moving left or right which I will need to fix, basically all I want to do is to increase the self.counter only when i press the left or right keys

```
52          if key[pygame.K_LEFT]:
53              checkx -= 3.5
54              self.counter += 1
55          if key[pygame.K_RIGHT]:
56              checkx += 3.5
57              self.counter += 1
58
59          #handling the animation
60
61          if self.counter > sprite_handler:
```

- To fix the problem i removed the self.counter += 1 from the handling animation part to only when the left or right keys are pressed making it so that the animation only starts if move the player left or right
- However a different problem I'm facing now is that when i stop pressing the left or right buttons the player sometimes stops mid animation which doesn't look good

```
58          if key[pygame.K_LEFT] == False and if key[pygame.K_RIGHT] == False:
59              self.counter = 0
60              self.index = 0
61              self.image = self.images_right(self.index)
```

- Here i make so that if im not pressing the left or right keys the counter reset and goes back to the first picture

```
27              self.images_left = []
```

```
img_left = pygame.transform.flip(img_right, True, False)
```

```
42              self.direction = 0
```

- Right now when i move left the player still faces right and to fix that i just need to create the same things as i did for the right and do it for the left
- I will use a variable called self.direction to control whether im facing left or right

```
# checks if left key is being pressed
if key[pygame.K_LEFT]:
    checkx -= 2 # moves player to left by this velocity
    self.counter += 1 # increses the counter the control the animation
    self.direction = -1 # direction changes to -1

# checks if right key is being pressed
if key[pygame.K_RIGHT]:
    checkx += 2 #] moves the player to the right by this velocity
    self.counter += 1  # increases the counter to control the animation
    self.direction = 1 # direction changes to 1
```

- I then add in the self.direction when i press the left and right keys and make it equal to 1 and -1

```
# selects the needed image based on which direction the player is facing
if self.direction == 1:
    self.image = self.images_right[self.index]
if self.direction == -1:
    self.image = self.images_left[self.index]
```

- Then here I select that if it's equal to 1 show the player right and if it's -1 show the player moving left

- However now when I press left and the player moves left when I let go my player goes back to facing right which i don't want

```
79              if self.direction == 1:
80                  self.image = self.images_right[self.index]
81              if self.direction == -1:
82                  self.image = self.images_left[self.index]
```

- So I copy paste it into the class section as well so that when i stop moving left my character keeps facing left until i press the right key
- Reviewing this milestone , I can say that i have achieved it and completed the objective as now my character can be seen moving through the animation when the left or right or up arrow keys are pressed

Implementing Collision - 20/08/2024

- The aim of this is to implement collision so that the character cannot leave the screen through the tiles. To do this I will split it into x and y - axis based collisions so that if the character cannot go further down due to a tile in the way he will still be able to move left or right on that tile

```
world = Game_data(game_data) # creating an instance by passing in the level layout
```

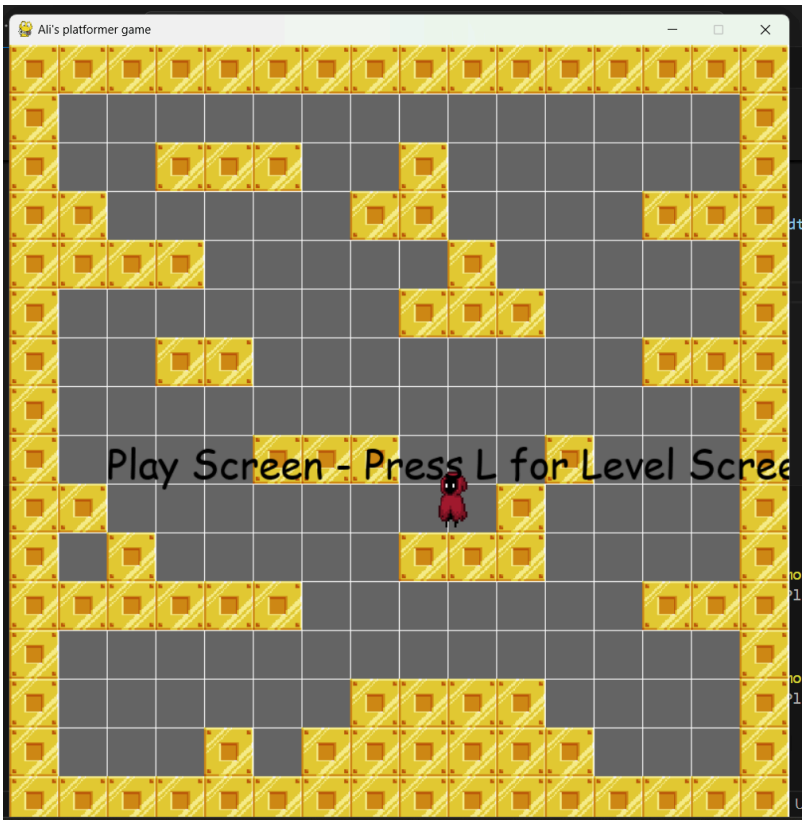
```
self.width = self.image.get_width() # set width of the sprite
self.height = self.image.get_height() # set height of the sprite
```

```
# collision Checker
for tile in world.tile_list:
    # x - axis collision
    if tile[1].colliderect(self.rect.x + checkx, self.rect.y, self.width, self.height):
        checkx = 0 # stops horizontal movement if there is collision
```

- To implement collision into my game i will be using pygame's built in collision detection system. I will be using a for loop to loop through all of the tiles in my game. Previously i created an instance of my world and so i have access to any of the variables in the world class and i can access them using world."whateveriwantoaccess" . I will need to separate the x and y collision detection since if the character collides with a tile by standing on I still want him to be able to move in the x direction. The rectangle data was stored in index 1 and therefore I used 1. I then use the in built collision detection to check whether the character collides with a tile but i do not use the actual character rectangle. I instead use a new rectangle which is a little shifted towards the original character's rectangle. I do this because if i use the actual player's rectangle with this collision detection system once I get a collision alert it will be too late because they've already collided. Therefore i create a new rectangle which is a little outside the boundaries of the original player's rectangle and I use that to create some space between the original character and the tile to get an alert of the collision with the tile and the new rectangle so that I can stop the original character in time before they collide

```
95              if tile[1].colliderect(self.rect.x, self.rect.y + checky, self.width, self.height):
96                  if self.vel_y < 0:
97                      checky = tile[1].bottom - self.rect.top
98                  if self.vel_y >= 0:
99                      checky = tile[1].top - self.rect.bottom
```

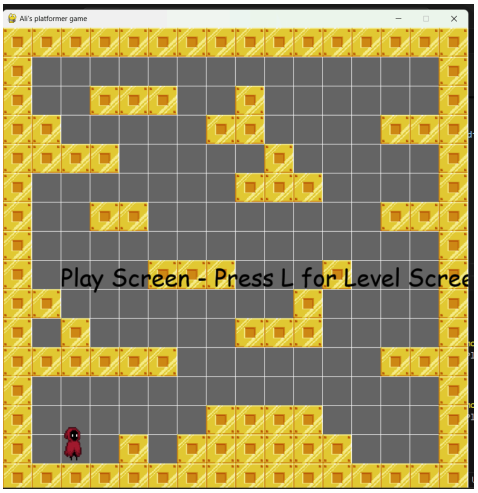
- To start with I will do a collision based on whether the character hits his head on the bottom of a tile. I write that if his y velocity is less than 0 (meaning that he has just jumped and is moving up) and there is a tile above him, then the maximum distance he can move is the distance between the bottom of the tile and his head
- I then do the same thing but the opposite for when the character lands on a tile. If his y velocity is greater than 0 (meaning he has moved down) the distance between him and the tile is the distance between the top of the block and the bottom of the character



Problem 4, Player Teleporting - 22/08/2024

- The player now stands on the tiles fine however when i jump under a block the player teleports to the top of the block he was under which isn't good, I now will look through my code to see how I can fix this

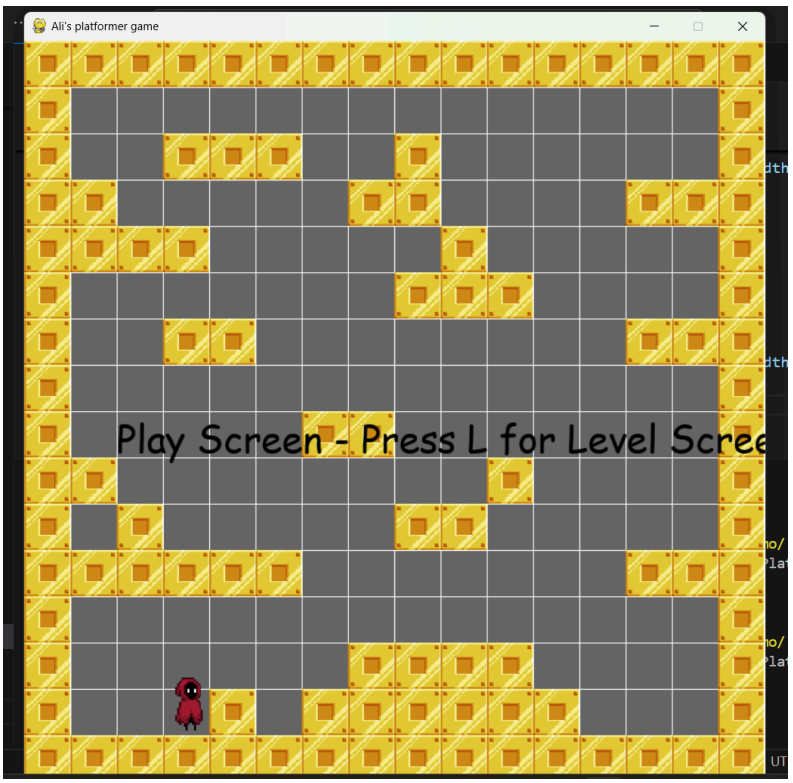
```
# y - axis collision
if tile[1].colliderect(self.rect.x, self.rect.y + checky, self.width, self.height):
    if self.vel_y < 0:
        checky = tile[1].bottom - self.rect.top
        self.vel_y = 0 # if player's head aligns with tile's bottom, stop the jump
    elif self.vel_y >= 0:
        checky = tile[1].top - self.rect.bottom
        self.vel_y = 0 # if player's bottom aligns with tile's top, stop the movement
```



- I now fixed the earlier problem by trying different things but the only thing that worked was adding in self.vel_y = 0 at the end of each if statement. Now when i press jump under a block it doesn't teleport to the top of the block I was under and instead the character just bangs his head with the bottom of the block and goes back down normally

```
103         # x - axis collision
104         if tile[1].colliderect(self.rect.x + checkx, self.rect.y, self.width, self.height):
105             checkx = 0
```

- Now collision in the x - axis is very easy as all i need to do is have the same first line as the y axis collision but instead of adding a new rectangle shifted a little outside the original player's rectangle in the y axis, i do it in the x axis and if there is a collision with that outside rectangle i then set the x velocity to zero so that the character cannot move anymore



- Now when I want to move left or right and there is a tile in the way the character just cannot move and will stay there until the user jumps over the block
- Reviewing the milestone, I have completed it and achieved the objective as I now have successfully implemented collision in to my game and the character stops moving when needed to

Adding Enemies/Obstacles - 23/08/2024

- The objective of this milestone is to add a slime - like monster into my game, I am planning on making the monster move left and right as this will increase the complexity of the level and make it more fun

```
# class for enemies
class enemies(pygame.sprite.Sprite):
    def __init__(self, x, y):
        pygame.sprite.Sprite.__init__(self) # initialising
        self.image = pygame.image.load('slimer1.png') # load image for slime enemy
        self.rect = self.image.get_rect() # create rectangle for image
        self.rect.x = x # set x position
        self.rect.y = y # set y position
```

- To create an enemy I created an enemies class and I will be using the sprite classes prebuilt into pygame. I then add in the constructor which will take in the x and y positions. I basically want my enemy to inherit some of the attributes of the built in enemy class. I then load in the image and create it a rectangle.

```
141                 if tile == 3:
142                     slimeguy = enemies(col_count * tile_size, row_count * tile_size)
```

- Then in the game_data class i make it so that in the level grid the number three means the slime enemy, basically making an instance of the monster

```
188    slimeguy_group = pygame.sprite.Group
```

  - Then using pygame's built in feature i make a group that i can add enemies into

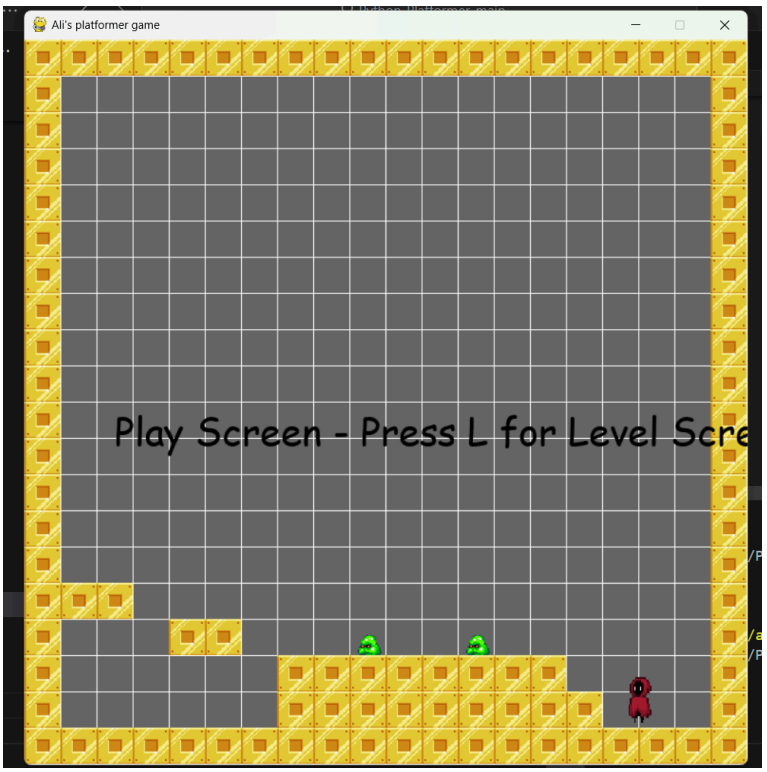
```
# checks if a 3 is present (slime enemy)
if tile == 3:
    slimeguy = enemies(col_count * tile_size, row_count * tile_size) # creates enemy object
    slimeguy_group.add(slimeguy) # adds to enemy group
```

- I then add to the group the monster character I created

```
182    [1,0,0,0,1,1,0,0,0,3,0,0,3,0,0,0,0,0,0,1],
```

```
249            slimeguy_group.draw(window)
```

- I then draw the monster onto the screen so that he can show and i add in the number 3 to the level grid where i want him to show



- Now as you can see the monster slime does show meaning everything is working fine and and he is placed perfectly on the blocks where i want him to be
- I've also remodelled/resized the map as i don't like how it was before, i made the tile's smaller so that i can add more functionality to the map and i will add in the rest of the tiles later

- Now if i leave them still like how it is now then the game won't be challenging at all and so therefore i am going to try to make them move a little right and left so that the user has to try jumping over them while they are moving

```
        self.slime_move = 1 # set speed for slime enemy
        self.slime_counter = 0 # tracks movement distance

    def update(self):
        self.rect.x += self.slime_move # updates the slime's x position
        self.slime_counter += 0.3 # slowly increase the movement
        if abs(self.slime_counter) > 40: # after  a certian distance chnage direction
            self.slime_move *= -1 # move opposite direction
            self.slime_counter *= -1 # chnage the movement counter
```

```
254                 slimeguy_group.update
```

- To make the slime character move i make the update function and increase the x position by 1 until it reaches 50 and once it does i make it flip and go in the opposite direction to come back , then i reset it back to zero
- I then update the sprite group so that it can show what's happening

```
def update(self):
    self.rect.x += self.slime_move # updates the slime's x position
    self.slime_counter += 0.3 # slowly increase the movement
    if abs(self.slime_counter) > 40: # after  a certian distance chnage direction
        self.slime_move *= -1 # move opposite direction
        self.slime_counter *= -1 # chnage the movement counter
```

- Before changing the code they were only going to the right a bit then going back to their original position but now i set the slime counter to -1 so that once it reaches 50 it becomes negative and starts counting to 0 making it go left then it switches back and forth between positive and negative so that the slime monster goes right then back to original position then a little left the back to original position and keeps repeating that
- Reviewing the milestone, I have achieved the goal and I now have a slime monster within my level which has made my game much more fun and challenging

Adding Enemies/obstacle - 23/08/2024

- The aim of this milestone is to incorporate another enemy which will be a spike. To do this I will add in an image of a spike and place it on specific tiles and so if the character collides with it he loses a live

```
# checks if a 4 is present (spike trap)
if tile == 4:
    Spike = spike(col_count * tile_size, row_count * tile_size + (tile_size // 2)) # creates a spi
    spike_group.add(Spike) # adds spike to spike group
```

- I want to add spikes in a certain area of the game and to do this i need to assign it a number so that the code knows this number stands for the spikes

```
# class for spike trap
class spike(pygame.sprite.Sprite):
    def __init__(self, x, y):
        # initialising
        pygame.sprite.Sprite.__init__(self)
        image = pygame.image.load('spike.png') # load spike image
        self.image = pygame.transform.scale(image, (tile_size, tile_size // 2)) # transform image to

        self.rect = self.image.get_rect() # create rectangle for image
        self.rect.x = x # set x position
        self.rect.y = y # set y position
```

- I then create class for the spikes and use similar code from the monster class, i load in the image and then scale it, i create a rectangle that i can use in the future for collision and assign the rectangle's position

```
spike_group = pygame.sprite.Group() # creating a sprite group for my spikes
```

- I then make a group for the spike so that i can add my spike to it

```
spike_group.add(Spike) # adds spike to spike group
```

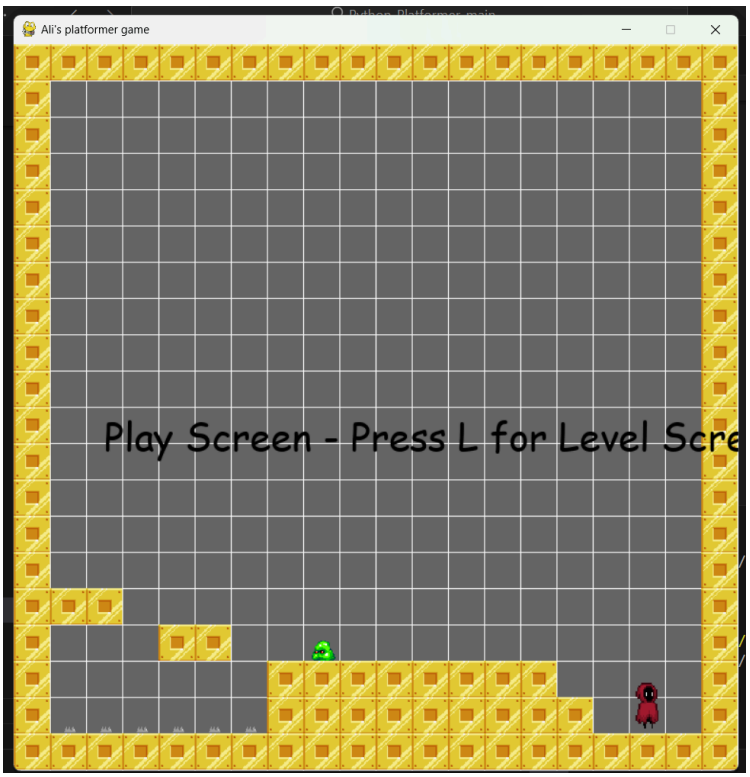
- Here i add my spike to the spike group

```
[1,4,4,4,4,4,4,1,1,1,1,1,1,1,1,1,0,0,0,1],
```

- I also adjust the game_data and place a 4 where i want the spikes to show

```
# draws what is needed to game window
slimeguy_group.draw(window)
spike_group.draw(window)
```

- Then i draw the spike to the window



- In the bottom left of the screen you can now see the spikes
- Reviewing this milestone, I can say that i have completed it and achieved the objective as you can see in the picture above

Collision With Enemies/obstacle - 24/08/2024

- The objective of this milestone is to make it so that when the character makes a collision with the slime monster or spikes the game freezes for now indicating losing.

```
13    end_game = 0
```

First i create a variable called end_game which will decide whether or not the game ends based on if the player makes contact with the monster or spikes

```
# checks for collision with slime enemy
if pygame.sprite.spritecollide(self, slimeguy_group, False):
    end_game = -1 # if collision is detected, one live is lost

# checks collision with spike trap
if pygame.sprite.spritecollide(self, spike_group, False):
    end_game = -1 # if collision detected, one live is lost
```

- I make it so that if the sprites collide with anything in the spike group or slime group the end_game variable becomes -1 indicating that the game needs to end

```
47      def update(self, end_game):
```

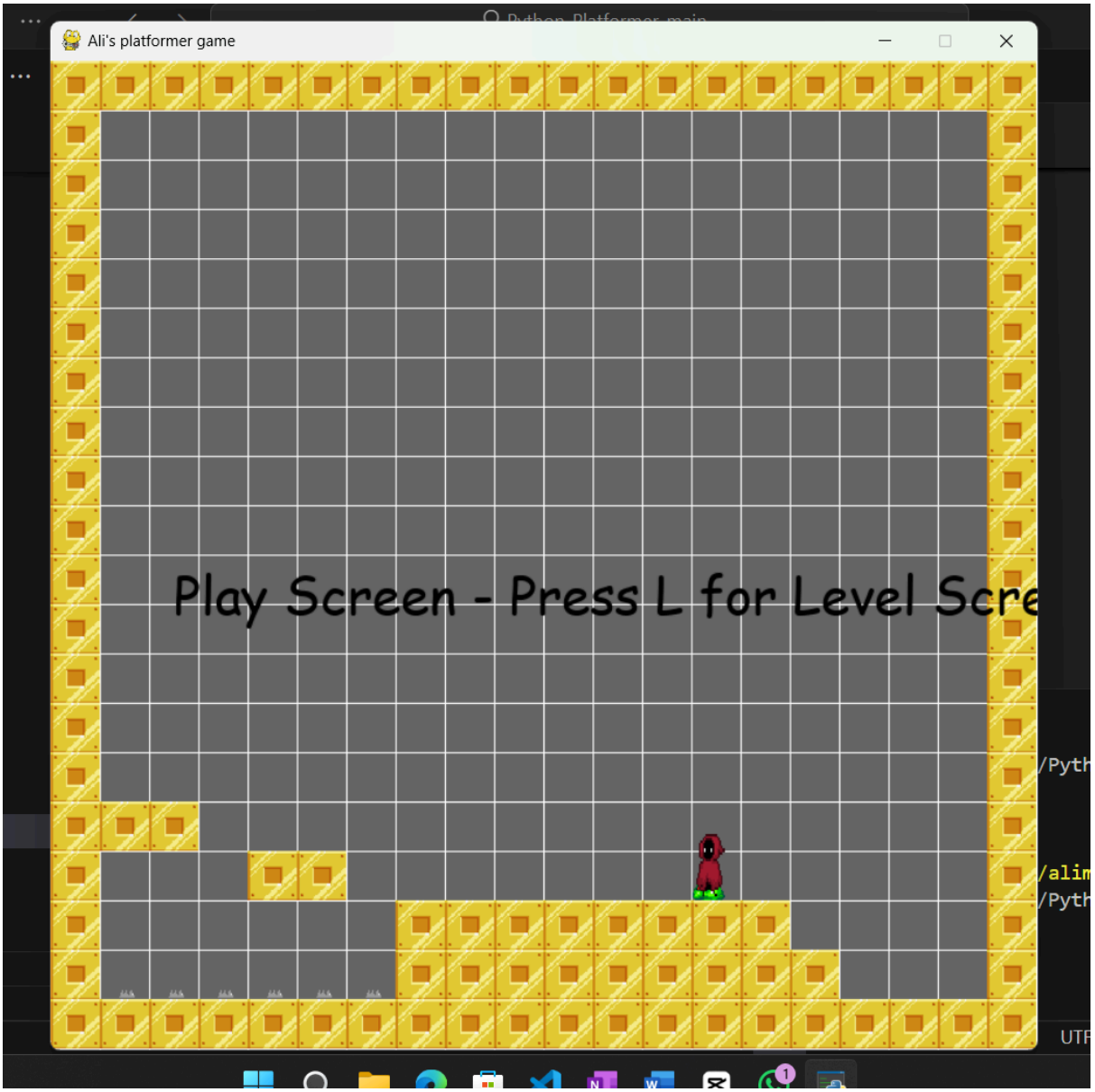
```
53          if end_game == 0
```

```
131         return end_game
```

- In the update function of the player class i add in end_game since it's a global variable and i indent everything in the update function so that i can add in and if statement if end_game is zero then everything can run

```
288         end_game = player.update(end_game)
```

- Here i make it so that the end_game variable is fed into the update method but since it's being returned i make it equal to the player update



- Now what happens when i collide with the monster if that my character freezes and i can't move him anymore but everything else continues just fine

```
# only updates if game is not closed
if end_game == 0:
    slimeguy_group.update()
```

- Now this part of the code makes it so that if collision occurs then the slime monster stops moving
- Reviewing the milestone, I can say that I have somewhat achieved it as the game now freezes however nothing else happens and i need to work on that

Adding Lives - 27/08/2024

- The aim of this milestone is to add in lives into the game so that the user has multiple attempts at the level before being forced to restart again, This makes the game more enjoyable,

```
275    def play_screen():
276        run = True
277        end_game = 0
278        lives = 5
279        player_start_x, player_start_y = 650, HEIGHT - 130
```

- In the beginning of the play screen function i set the lives equal to 5 and I also set the players position so that i can use it to reset the player once he collides with the monster or spikes

```
# handles lives and game over
if end_game == -1:  # if player loses
    lives -= 1  # decrease lives by one
    if lives > 0: # if player has not lost all lives
        player.rect.x, player.rect.y = player_start_x, player_start_y # spawn player again
        end_game = 0  # rests
    else:
        return GAME_OVER, score # if no lives are left, go to game over state
```

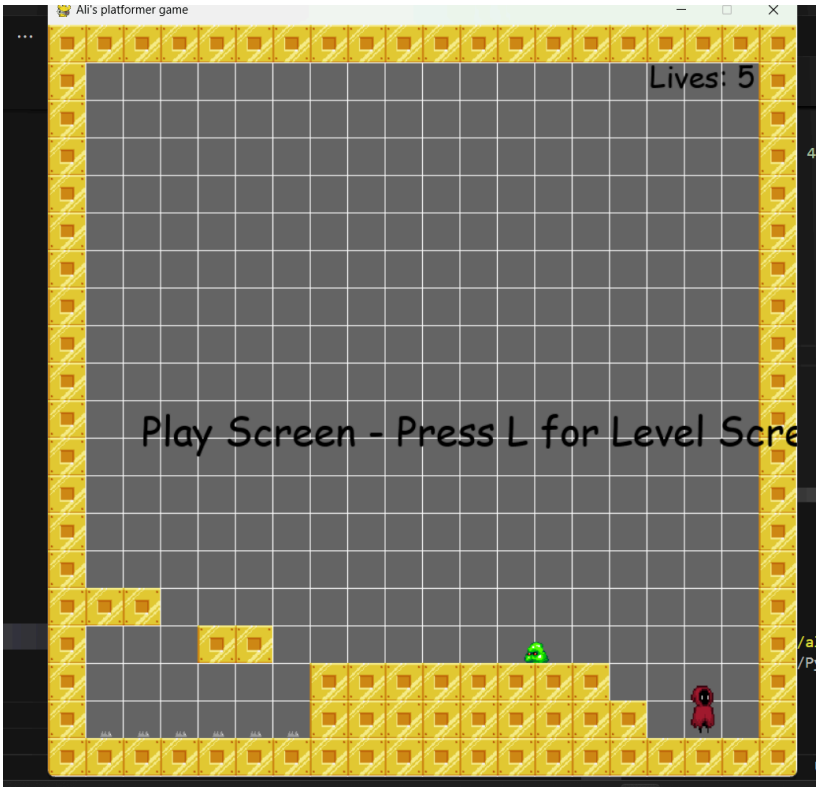
- Also in the play screen function i add in that if collision is detected and end_ game is equal to -1 to reduce the lives by 1. As long as lives is bigger than 0 then the game will keep resetting the player to th beginning once he collides and once the lives are all used the game will go to the game over screen

```
304            draw_text(f"Lives: {lives}", 30, BLACK, WIDTH - 150, 30)
```

- I also display the lives so that the user knows how many chances he has left

```
for event in pygame.event.get(): # checks for events
    if event.type == pygame.QUIT:
        run = False # If user wants to quit , it will stop running
    if event.type == pygame.KEYDOWN: # checks if any keys are pressed
        if event.key == pygame.K_RETURN: # checks if enter key is pressed
            return START # outputs the main menu
        elif event.key == pygame.K_ESCAPE: # if the escape key is pressed
            pygame.quit() # exits the game
            return START
```

- I then modify the game over function to make it so that the user can restart by pressing enter and leaving the game by pressing escape



- Now as you can see the lives are displayed in the top right corner and the player resets every time he collides, the game also goes to the game over screen once the lives run out
- Reviewing the milestone I can say that the objective has been achieved, this is because lives are fully functioning in my game allowing the user to have 5 failed attempts before having to restart the game

Adding A Score System - 27/08/2024

- The aim of this milestone is to add in a score system which increases based on how long the user stays alive for. This makes the game more challenging and allows for some competitiveness between different players

```
score = 0  # score starts form zero
score_update_time = pygame.time.get_ticks() # timer for score
```

- I first initialise the score and use pygame's built in feature of increasing the score based on how much time has passed

```
# updates score every second
current_time = pygame.time.get_ticks()
if current_time - score_update_time > 1000:
    score += 1 # increases score by one each second
    score_update_time = current_time
```

- This part of the code increases the score every 1 second by 1 which i what I want as the game will include multiple levels and so if the score is increasing too fast and will become a very large number

```
311         draw_text(f"Lives: {lives}", 30, BLACK, WIDTH - 150, 30)
312         draw_text(f"Score: {score}", 30, BLACK, 50, 30)
```

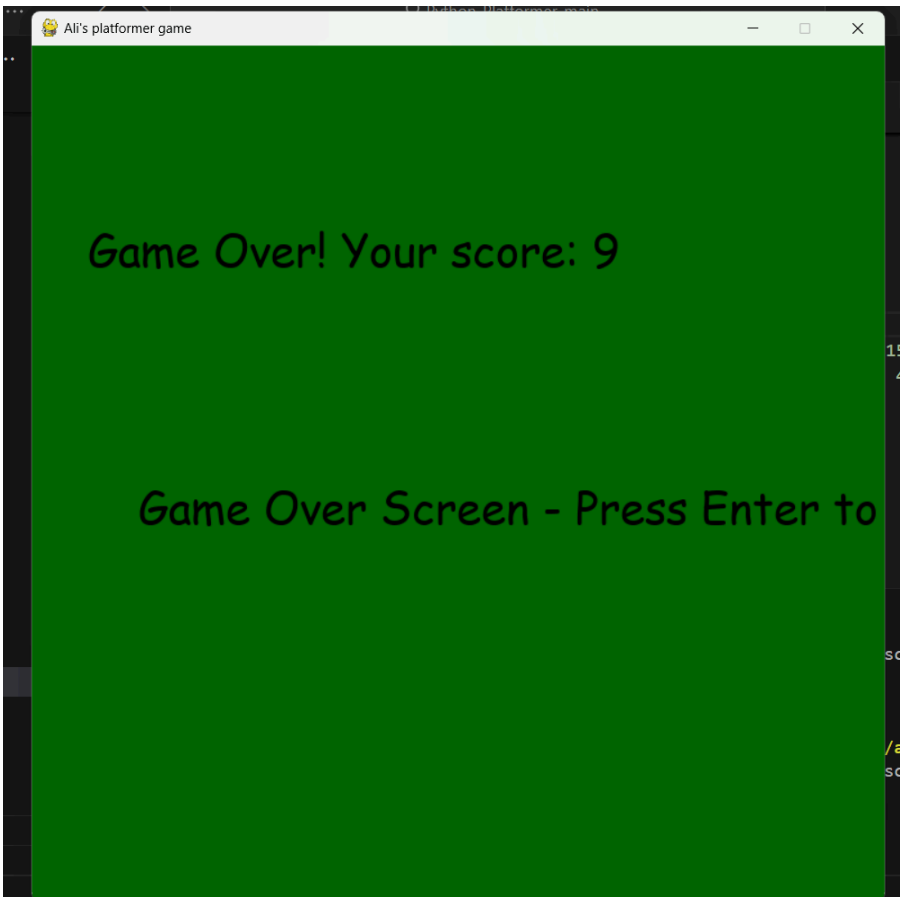
- I then draw the score under where I drew the lives

```
# main game function
def main():
    clock = pygame.time.Clock() # helps control the game's FPS
    state = START  # program starts in the start state
    score = 0
```

- I initialise the score in the main function as well to store to be able to store the score in different game states

```
398         elif state == PLAY:
399             state, score = play_screen()
400         elif state == GAME_OVER:
401             state = game_over_screen(score)
```

- This part of the code here helps with passing the score through the game state's



- Now as you can see the score is displayed once i lose all my 5 lives
- Reviewing this milestone , I have achieved it and completed the objective since as you can see in the image above , the scores keeps increasing as long as the character is alive and then is outputted once all 5 lives are lost

Creating The Second Level - 2/09/2024

- The objective of this milestone is to add in the second level for my game. To do this I will use the same technique of using a grid with different numbers representing different objects but change it up so that it's not the same level as the first

```
game_data_2 = [
        [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,1,1,1],
        [1,0,1,0,0,1,0,0,1,0,0,1,0,0,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,3,0,0,0,0,1,0,0,0,1],
        [1,4,4,4,4,4,4,4,1,1,1,1,1,1,1,1,1,1,0,0,1],
        [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
    ]
```

- To create the second level i created another grid like for the first level which contains where each of the tiles will go

```
# checks if level finished
if world.transition_block and world.transition_block[1].colliderect(self.rect.x, self.rect.y, self.width, self.height):
    return "next_level" # if player collides with the transition door, move to next level
```

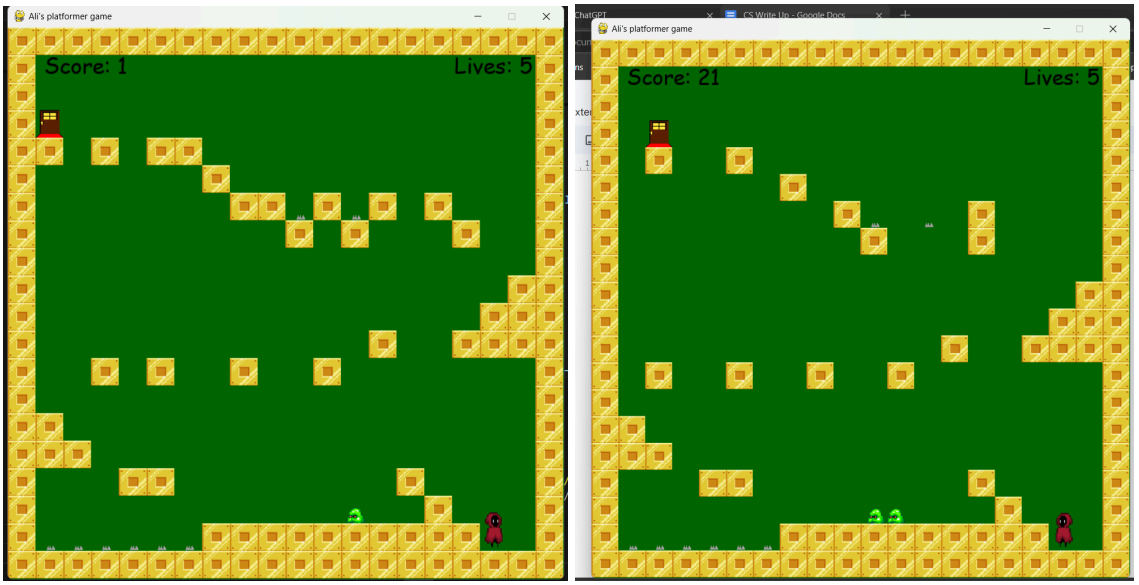
- I then add an if statement which will check if my player collides with the transition block, this block will be responsible for moving to the next level. It will return “next level” if this collision has occurred.

```
# checks for level transition
if end_game == "next_level":


    current_level = game_data_2  # load level 2 data
    world = Game_data(current_level)  # creates world for second level
    player.rect.x, player.rect.y = player_start_x, player_start_y  # sets player's spawn point for second level
    end_game = 0 # changes the state
```

- Using the above code, once “next level” has been returned the current level will be changed to the second level and the world will be updated with this second level, the player’s position will also be updated and end_game will be reset to zero to show that the game is on

```
# checks if a 5 is present(transition door)
if tile == 5:
    img = pygame.image.load('door.png') # load the image
    img = pygame.transform.scale(img, (tile_size, tile_size)) # resize image
    img_rect = img.get_rect() # create rectangle for image
    img_rect.x = col_count * tile_size # set x position
    img_rect.y = row_count * tile_size # set y position
    self.transition_block = (img, img_rect) # store the transition door
```

- In the for loop that checks each row in the grid I also have to add the transition block which will be a door picture. I load the image and size it up to the tile size and name the transition block



- As you can see now the door has been loaded and once the player reaches it and makes contact with it the second level is loaded

Problem 5, Enemy Duplication - 04/09/2024

- A problem i'm facing now is that the slime monster from level 1 is not going away and meaning in level 2 i have 2 monsters even i only want one. To fix this i think i need to remove all of them when level 1 is finished and put them back again in level 2

```
# checks for level transition
if end_game == "next_level":
    slimeguy_group.empty()  # clear enemy group
    spike_group.empty() # clear spike group
```

```
world = Game_data(current_level)  # creates world for second level
```

- Here i empty the groups for the sprites when the level transitions to the next and I make sure the second level only loads its contents not the one before as well
- Reviewing the milestone I can now say that I have achieved what I intended to do as now once the player makes contact with the transition door they automatically get spawned into the next level allowing for and easy and quick transition making the game look good

Adding New Blocks/Tiles For Level 2 - 01/10/2024

- The objective of this milestone is to change up the look of the second level so that each level look unique which in turn makes the game look good and feel more fun

```
# Function for levels
class Game_data():
    def __init__(self, data): # Constructor taking in self and data as arguments
        self.tile_list = [] # creating a list
        gold_img = pygame.image.load('gold.png') # loading in the gold tile image
        silver_img = pygame.image.load('silver.png') # loading in the silver tile image
```

- In my class for the game_data i load in the image and save it as silver_img

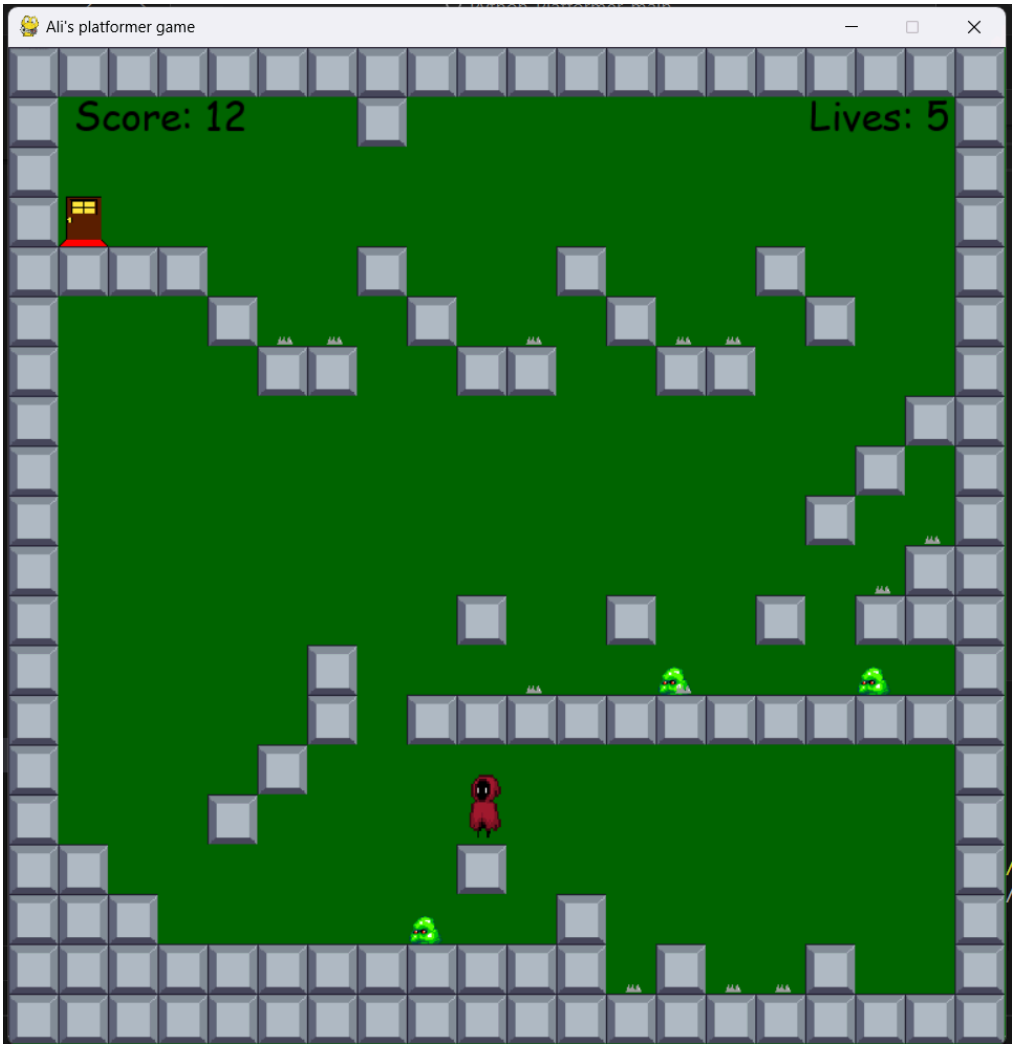
```
    # checks if a 6 is present (silver tile)
    if tile == 6:
        img = pygame.transform.scale(silver_img, (tile_size, tile_size)) # resize image
        img_rect = img.get_rect() # create a rectangle for image
        img_rect.x = col_count * tile_size # set x position
        img_rect.y = row_count * tile_size # set y position
        tile = (img, img_rect) # create a tuple for image and rectangle
        self.tile_list.append(tile) # add tiel to tile list
```

- I then reuse some of the code i used before and change the tile number to 6 and change the image i want to use to the silver image

```
game_data_2 = [
        [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
        [1,0,0,0,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,1,1,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1],
        [1,0,0,0,1,4,4,0,1,0,4,0,1,4,4,0,1,0,0,1],
        [1,0,0,0,0,1,1,0,0,1,1,0,0,1,1,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,4,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,1,1,1],
        [1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,1,1,1],
        [1,0,0,0,0,0,1,0,0,0,4,3,0,4,0,3,0,0,0,1],
        [1,0,0,0,0,0,1,0,1,1,1,1,1,1,1,1,1,1,1,1],
        [1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1],
        [1,1,1,0,0,0,3,0,0,0,0,1,0,0,0,0,0,0,0,1],
        [1,1,1,1,1,1,1,1,1,1,1,1,4,1,4,4,1,0,0,1],
        [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
    ]
```

```
game_data_2 = [
        [6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6],
        [6,0,0,0,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,6,6,6,6,0,0,0,6,0,0,0,6,0,0,0,6,0,0,6],
        [6,0,0,0,6,4,4,0,6,0,4,0,6,4,4,0,6,0,0,6],
        [6,0,0,0,6,6,6,0,6,6,0,0,6,6,6,0,0,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,0,4,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,6,6,6],
        [6,0,0,0,0,0,0,0,0,0,6,0,0,6,0,0,6,6,6,6],
        [6,0,0,0,0,0,6,0,0,0,4,3,0,4,0,3,0,0,0,6],
        [6,0,0,0,0,0,6,0,6,6,6,6,6,6,6,6,6,6,6,6],
        [6,0,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,6,0,0,0,0,0,0,6,0,0,0,0,0,0,0,0,0,0,6],
        [6,6,6,0,0,0,3,0,0,0,0,6,0,0,0,0,0,0,0,6],
        [6,6,6,6,6,6,6,6,6,6,6,6,4,6,4,4,6,0,0,6],
        [6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6],
    ]
```

- I then change the level data and change all of the number 1's to number 6's, this will change all of the gold blocks to silver blocks

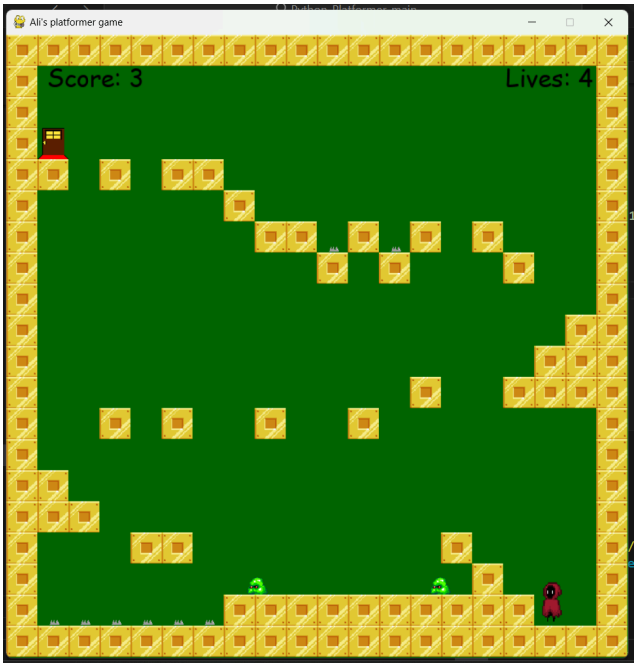


- As you can see now my level 2 has changed the gold blocks to silver blocks for the entire level and i have achieved this milestone

- Reviewing the milestone, I can say I have achieved the objective since the second level's tiles are different and I also intended to change the background colour which will make the game look completely different

Problem 6, Enemy Duplication Again - 10/10/2024

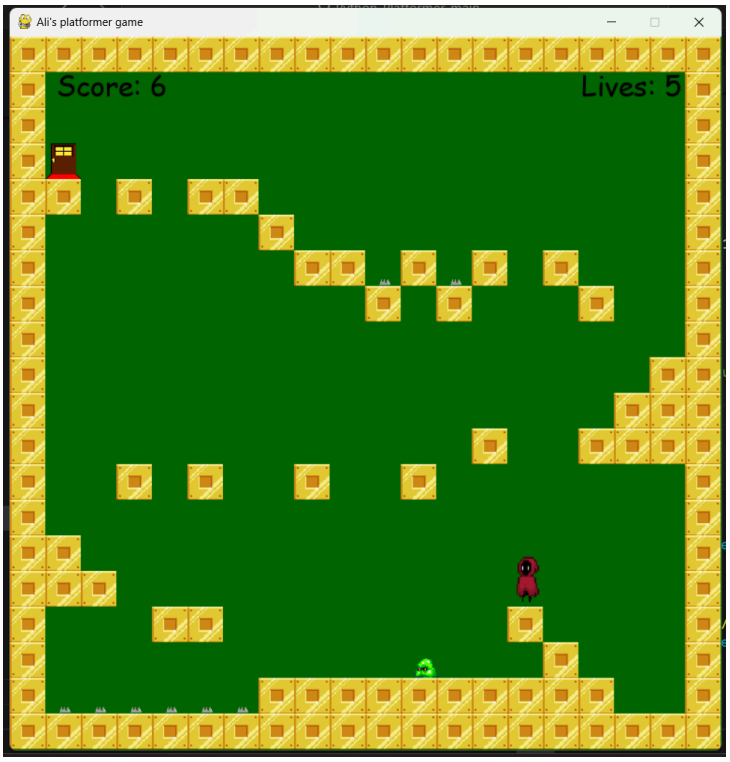
- A problem which I had is that when i was testing my game, if I lose all my lives within level 1 , it will send me to the game over screen which is fine however when i start the game again it spawns and duplicates the enemies again



- As you can see there are two monsters in level 1 when there should only be one, this is because the sprite group does not get cleared after all lives are lost.
- To fixc this i will empty the sprite group when the game over screen is shown so when they player decides to play again the enemies will not duplicate and will remain normal

```
slimeguy_group.empty()
spike_group.empty()
# this clears the sprite group so that they don't duplicate when user loses all lives
```

- Here i empty the sprite group in the game over screen function so that they do not duplicate



- Here as you can see even when i lose all my lives and restart, the enemies remain normal and do not duplicate and so therefore i have fixed the problem

Creating Level 3 - 12/10/2024

- The objective of this milestone is to create the third level for my game. This is most likely going to be the final level in the game, however I may later add more levels if I wanted to later

```
# checks for level transition
if end_game == "next_level":
    slimeguy_group.empty()  # clear enemy group
    spike_group.empty() # clear spike group

    current_level = game_data_2  # load level 2 data
    world = Game_data(current_level)  # creates world for second level
    player.rect.x, player.rect.y = player_start_x, player_start_y  # sets player's spawn point for second level
    end_game = 0 # changes the state
```

- Currently my level transition is looking like this. Once end game outputs “next level”, the current level is set to the second level’s data and is outputted. This currently doesn’t allow

me to add a third level since I do not have a variable keeping track of which level I am on. Therefore I will be creating a variable which will keep track of the levels and it will automatically increase once “next level” is outputted

```
level_number = 1 # variable to keep track of levels

# function for playing state
def play_screen():
    global world, level_number # helps accessing the global variable "world"
```

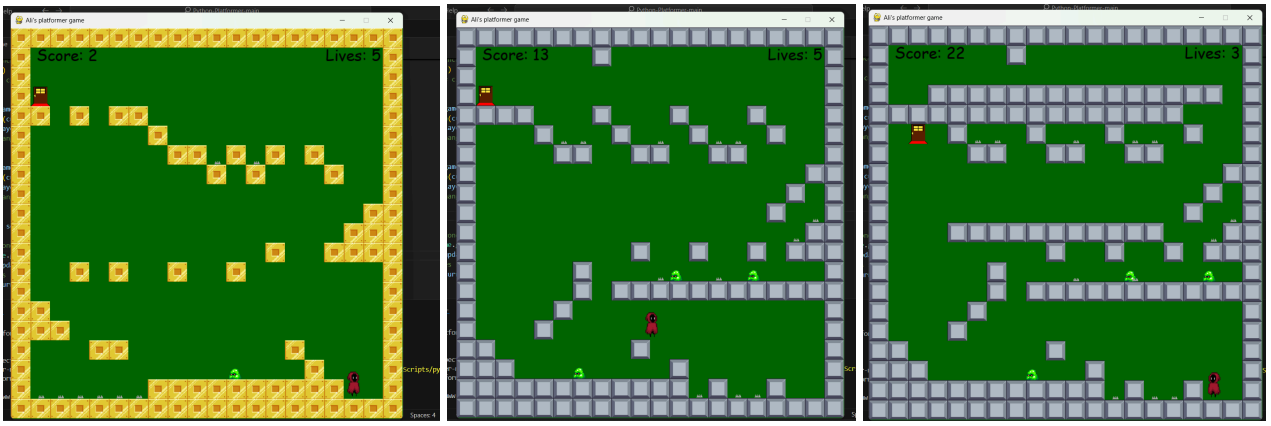
- Here I set the level_number variable to 1 meaning that the first level will be displayed first. I then pass it in as a global variable in the function

```
 # checks for level transition
 if end_game == "next_level":
     level_number += 1 # increases by one once "next level" is ouputted
     slimeguy_group.empty()  # clear enemy group
     spike_group.empty() # clear spike group
```

- I then increase it by one every time “next level” is outputted so that the next level can be displayed

```
if level_number == 2:
    current_level = game_data_2  # load level 2 data
    world = Game_data(current_level)  # creates world for second level
    player.rect.x, player.rect.y = player_start_x, player_start_y  # sets player's spawn point for second level
    end_game = 0 # changes the state

if level_number == 3:
    current_level = game_data_3 # load level 3 data
    world = Game_data(current_level) # creates world for third level
    player.rect.x, player.rect.y = player_start_x, player_start_y # sets player's spawn point
    end_game = 0 # changes the state

else:
    return GAME_OVER, score # if no more levels available, returns game over state
```

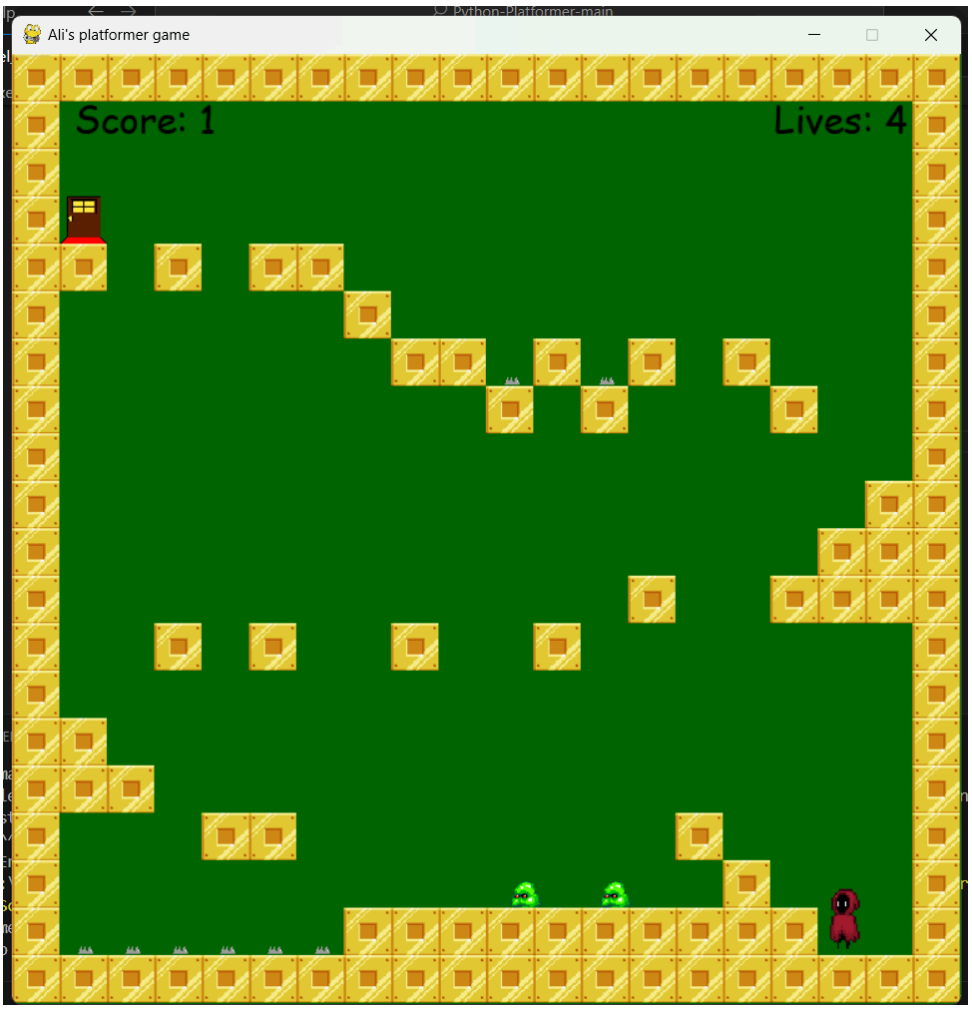
- I then write that if the variable “level_number” is equal to 2 or 3 then the needed data will be passed into “current_level” and displayed
- I also make it so that if no more levels are available (the user has completed all the level) then the game over screen is shown with their score



- As you can see here I go through all 3 levels and they are working. Level 3 has not been finished and I still have not changed all the tiles.
- Reviewing the milestone, I can say that I have achieved the desired objective, this is because I have now got a third level implemented into my game and it is fully functional

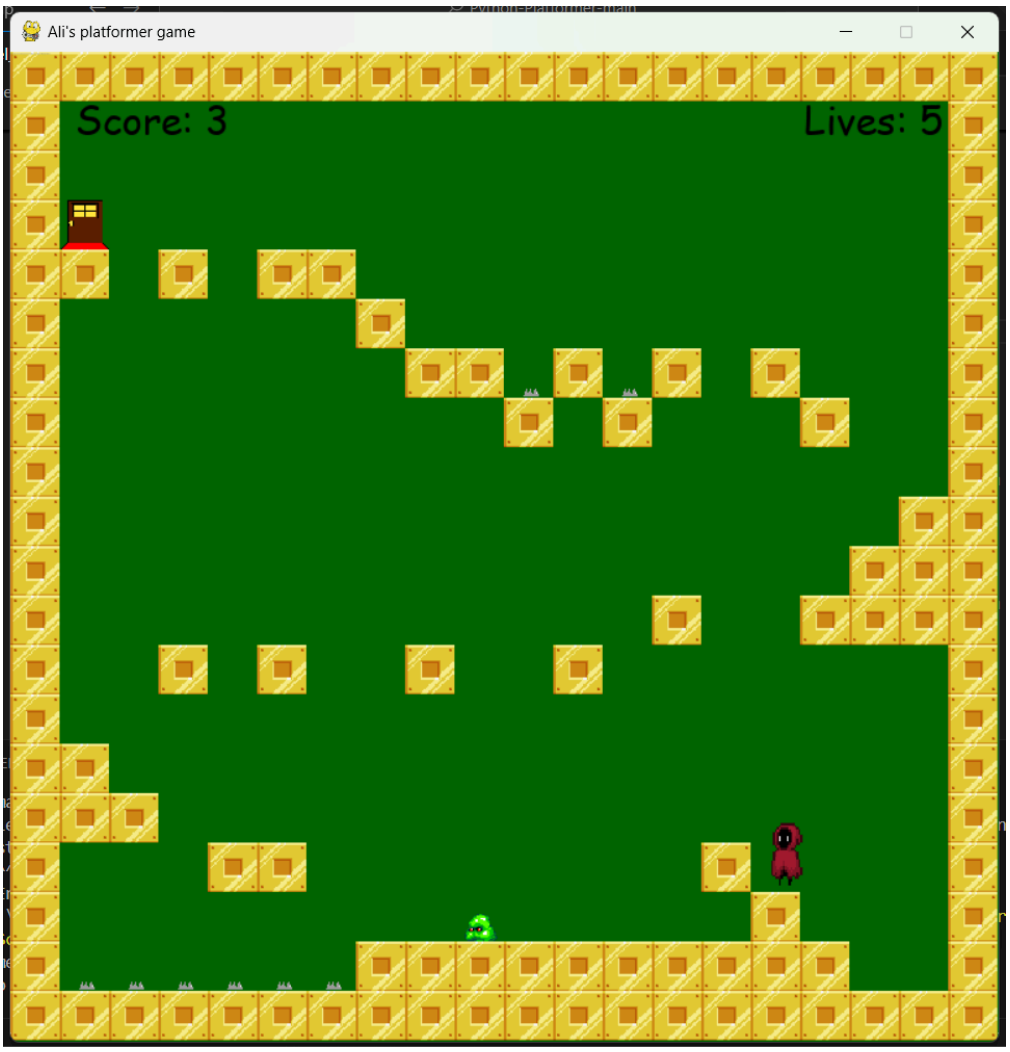
Problem 7, Enemy Duplication Again - 15/10/2024

- The aim of this milestone is to fix a problem I have where when the user loses all lives and attempts to play the game again , the slime monster from other levels spawn in as well as can be seen in the picture below




```
# handles lives and game over
if end_game == -1:  # if player loses
    lives -= 1  # decrease lives by one
    if lives > 0: # if player has not lost all lives
        player.rect.x, player.rect.y = player_start_x, player_start_y # spawn player again
        end_game = 0  # rests
    else:
        slimeguy_group.empty()  # clear enemy group
        spike_group.empty() # clear spike group
        player.rect.x, player.rect.y = player_start_x, player_start_y # spawn player again
        return GAME_OVER, score # if no lives are left, go to game over state
```

- After thinking about how I can solve this problem, I came to the conclusion that I need to empty the sprite group once the player all their lives and so I add this in the else section of the code as can be seen above



-
- Reviewing the milestone I can say I have achieved the desired output as now when the player attempts to play the game again after losing all their lives, the monster is not duplicated and is now working properly

Renaming Variables And Organising Code - 17/10/2024

- The aim of this milestone is to make sure all my variable names are appropriate and to make sure my code is easily readable

```
game_data = [
```

```
game_data_2 =
```

```
game_data_3 =
```

- These are right now the names of the grid layouts of my levels. I will be changing them to level1, level2 and level3 as I find them more appropriate and easier to follow. This will also mean I will have to change wherever in the code I used these names.

```
class Game_data():
```

- I'll also be changing Game_data to name it class levels and I will have to do that for wherever it is repeated again within my code

Implementing A Leaderboard - 25/10/2024

- The aim of this milestone is to add a leaderboard into my game. This is to add competitiveness into the game. I want it so that when the user loses all their lives the game asks for their name and saves their name with the score they achieved. I will then use a bubble sort to sort the leaderboard into the top 5 scores

```
# Function to save score to file
def WriteScore(name, score): # takes in arguments
    with open('data/PlayerScores.json', 'r') as file: # opens JSON file read mode
        data = json.load(file) # loads data of file into dictionary called data
    data[name] = score # updates the score in data
    with open('data/PlayerScores.json', 'w') as f: # opens JSON file in write mode
        json.dump(data, f) # updates file with new data
    print(data) # prints the updated data
```

- The purpose of this function is to read the file, update it with the new data, write the updated data back to the file and then display it. First I define the function and it will take in two arguments, the name and score, I then open the JSON file in read mode and load it into a dictionary called "data". I then update it with the new score and name. Then I open the JSON file again but now in write mode and give it the updated data, lastly I print the updated data.

```
name = "" # Store name as an empty string
```

- I create a variable called "name" and store it as an empty string for now and it will get updated when the user inputs their name and it will get saved

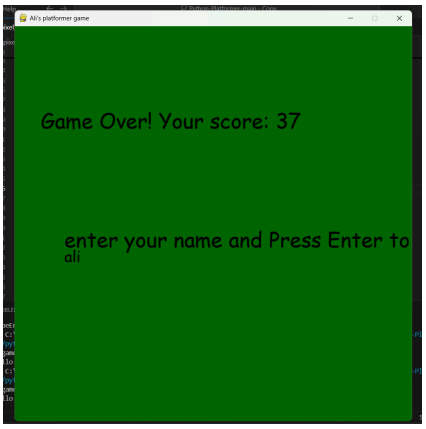
```
draw_text(f"Game Over! Your score: {score}", 40, BLACK, WIDTH // 15, HEIGHT // 5)
draw_text("enter your name and Press Enter to go to Start Screen", 40, BLACK, WIDTH // 8, HEIGHT // 2) # Displays the text
draw_text(name, 30, BLACK, WIDTH // 8, HEIGHT // 2 + 40) # shows written name
```

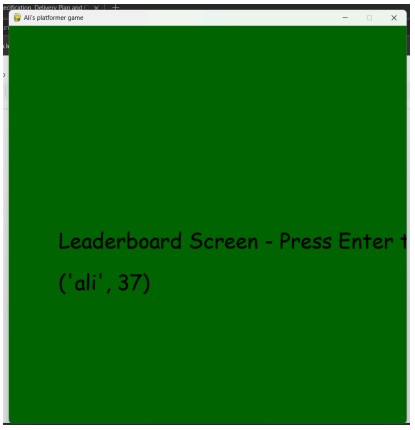
```
if event.key == pygame.K_RETURN: # checks if enter key is pressed
    WriteScore(name, score) # passes to the function the name and score
    return START # outputs the main menu
elif event.key == pygame.K_BACKSPACE:
    name = name[:-1]  # Remove last character
else:
    name += event.unicode # allows user to write their name
```

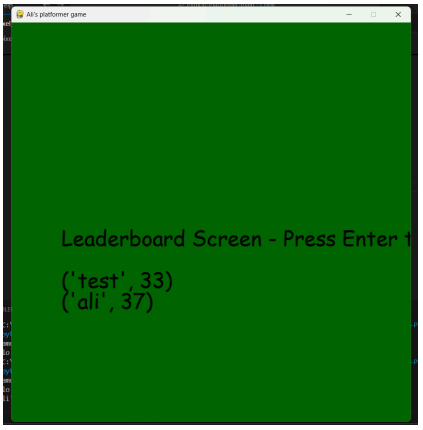
- Here in the game over screen I let the user know to enter their name so that we can save it with their score. Then I make sure when the user presses enter to go back to the main menu I give the function “writescore” the inputted name and score. I also allow the user to remove characters in their name incase they mispress a key

```
index = 1 # Keeps track of each score entry
with open('data/PlayerScores.json', 'r') as file: # Opens file in read mode
    data = json.load(file) # Loads file into dictionary
    data = list(data.items()) # changes dictionary into list of tuples
    for mx in range(len(data)-1, -1, -1): # Starts Bubble Sort Outer loop
        swapped = False # flag to detect any swaps
        for i in range(mx): # starts inner loop
            if data[i][1] < data[i+1][1]: # compares scores
                data[i], data[i+1] = data[i+1], data[i] # If sorting is needed, swaps the positions
                swapped = True # if sorting needed, changes flag to true
        if not swapped: # cehcks if no swaps happened
            break # stops loop

    for name in range(len(data)): # Rendering the names
        index += 1 # increments
        if index <= 6: # only renders top 5 scores
            draw_text("{}".format(data[name]), 40, BLACK, WIDTH // 8, HEIGHT // 2 + (index*40)) # draws text onto screen

pygame.display.update() # updates the screen
```

- The aim of this part of the code is to read the scores from the JSON file, sort them using a bubble sort and display the top 5 scores
- To do this I first set index = 1 to keep track of the ranks of each score entry. I then open the JSON file in read mode and load it into a dictionary called “data”. I then convert the dictionary into a list of tuples of name,score. I then use a bubble sort to sort the scores in ascending order from lowest to highest. Lastly I render the top 5 scores so that they can viewed







- In the left image above is when I complete level 3, it asks for my name so I write my name, then I press enter to go back to the main menu and press L to go to the leaderboard screen. As can be seen in the image in the middle, my name is written with the score I achieved. If i play again and get a better score then as can be seen in the image on the right, The bubble sort is working since it sorts from ascending order which is what I want

Alpha Testing

Menu Testing

Test No	What I test	Reason	Test Data	Valid/Invalid/Boundary	Expected Outcome
1	Starting the game	To see if the user will be able to load up the game to play it	Pressing start on visual studio code	Valid - Pressing start	Valid - Game will load and will first show the main menu
2	Loading level 1	To check if keys work in navigating through the game and allowing the user to start playing level 1	Pressing the ENTER key	Valid - Pressing ENTER Invalid - Pressing T	Valid - Level 1 will be loaded allowing the user to get started in playing the game Invalid - Nothing should happen
3	Loading leaderboard	To see if leaderboard	Pressing the L key	Valid - pressing L	Valid - Leaderboard

	screen	screen will be shown to allows users to see if they beat others or not		Invalid - pressing D	screen will be loaded allowing the user to see scores of previous players Invalid - Nothing should happen
4	Loading Help screen	To check if help screen will be loaded and if user will be informed on the important objectives of the game	Pressing the H key	Valid - Pressing H Invalid - Pressing A	Valid - Help screen will be loaded to inform user of important information of the game Invalid - Nothing should
5	Exiting the game	To make sure the user can exit the game once they are finished with all three levels or when they get tired	Pressing the escape key	Valid - Pressing esc Invalid - Pressing shift	Game will close
6	Going back to main menu from leaderboard screen	To check if the user will be able to navigate through the menus	Pressing ENTER key in leaderboard screen	Valid - pressing ENTER Invalid - pressing H	The game should go back to the main menu after ENTER key is pressed
7	Going back to main menu from help screen	To check if the user will be able to navigate through the menus	Pressing ENTER key in help screen	Valid - pressing ENTER Invalid - pressing L	The game should go back to the main menu after ENTER key is pressed
8	If game over screen loads when all lives are lost	To check if game knows when all lives have been lost	Losing all 5 lives using the arrow keys	Valid - Using arrow keys	Game should transition to the game over screen once all 5 lives are lost
9	Going from game over screen to the main menu	To check if the user can move to the main menu once they lose all	Pressing the ENTER key when in the game over	Valid - Pressing ENTER in the game over	Game should transition to the main menu

		their lives	screen	screen	

Character Testing

Test No	What I test	Reason	Test Data	Valid/Invalid/Boundary	Expected
10	If character can jump	To see if the character jumps as intended allowing the user to jump over obstacles and enemies when needed	Pressing the up arrow key when in playing the levels	Valid - pressing the up arrow key Invalid - pressing U	Character will jump a height of just over 1 tile to allow them to go through the level
11	If character can jump over the slime monster	To see if its too easy or too hard to jump over the slime monster	Pressing the up arrow and left arrow key to avoid the monster	Valid - pressing the up and left arrow Invalid - pressing U	Character should jump over the monster when monster gets too close
12	If character can move left and right	To check if user will be able to move character through the levels	Pressing and holding the left then right arrow key	Valid - pressing and holding the left then right arrow key Invalid - pressing T	Valid - Character should move left for a little then right Invalid - nothing should happen
13	If character appears	If character does not appear then the user will not be able to play the game	Pressing the ENTER key to load level 1	Valid - Pressing the ENTER key Invalid - Pressing W key	Valid - Level 1 should load with the character in the bottom right of the level Invalid - nothing should happen
14	If character falls through the tiles	If character falls through the tiles then there will be no character to move throughout the levels to	Pressing the ENTER key to load level 1	Valid - Pressing the ENTER key Invalid - Pressing the I key	Valid - Level 1 should load and the character should be standing on the golden tile without falling through them

		complete the game			Invalid - Nothing should happen
15	If character can jump onto other tiles	If the character cannot jump onto other tiles within the levels then the user will not be able to complete the game	Pressing the UP arrow key and LEFT arrow key at the same time	Valid - Pressing the UP arrow key and LEFT arrow key at the same time Invalid - Pressing X	Valid - Character should easily be able to jump onto another tile and stand on it Invalid - nothing will happen
16	If the character dies and resets to spawn position when they collide with a slime monster	To make sure that the game is somewhat challenging and is not too easy and that the collision system is working	Using the UP, LEFT and RIGHT arrow keys to move the character into the slime monster	Valid - Using the UP, LEFT and RIGHT arrow keys	Valid - When the character collides with the slime monster the character should respawn at the bottom right of the level
17	If the character dies and resets to spawn position when they collide with the spike trap	To make sure the collision system between the spike and the character is working	Using the UP, LEFT and RIGHT arrow keys to move the character onto the spike trap	Valid - Using the UP, LEFT and RIGHT arrow keys	Valid - When the character collides with the spike trap the character should respawn at the bottom right of the level
18	If the character stop moving up in their jump when they collide with a tile above them	To make sure the collision system between the character and the tiles working	Use the UP arrow key to make the character jump and hit their head on the bottom of a tile	Valid - Pressing the UP arrow key Invalid - Pressing the DOWN arrow key	Valid - Character stops jumping when the head of the character reaches the bottom of the tile Invalid - Nothing should happen
19	If the character stops moving when colliding with the tiles horizontally	To make sure the character cannot go off the screen and that the tiles stop them	Using the LEFT and RIGHT arrow keys to move the character into the tiles	Valid - Pressing the LEFT and RIGHT arrow keys	Valid - Character should stop moving when colliding with the tiles

Level System Testing

20	If level one loads when needed	Level one should load when the user wants to play the game	Clicking the ENTER key within the main menu	Valid - Pressing the ENTER key	Valid - Upon pressing the ENTER key the game should transition to level one
21	If level 2 loads once the character collides with the transition door in level 1	To allow the user to continue playing the game	Moving the character using the arrow keys into the transition door	Valid - Using the arrow keys to navigate the character	Valid - Game should transition to level 2
22	If level 3 loads once the character collides with the transition door in level 2	To allow the user to continue playing the game	Moving the character using the arrow keys into the transition door	Valid - Using the arrow keys to navigate the character	Valid - Game should transition to level 3
23	If level 1 load fine again once user completes all three levels and inputs their name for the leaderboard	To make sure game can be played again once completed	Pressing ENTER on main menu once game has been completed	Valid - Pressing ENTER	Valid - Game should load Level 1

Leaderboard/Score Testing

24	If timer works throughout all the levels or not	Timer is needed so that it can passed to the leaderboard with the name	Loading level 1 will start the timer	Valid - Clicking ENTER on the main menu to load level 1	Valid - Game should transition t level 1 and timer should start and remain for

		of the user			all 3 levels
25	If game only asks for your name input and saves your score after completing level 3 and not level 1 or 2	The game shouldn't take name input or pass the score if all 3 levels have not been completed	Losing all lives to see if game asks for name input and saves the score	Valid - Using the arrow keys to lose all 5 lives	Valid - Game should not ask for name input or save the scoreif level 3 has not been completed
26	If leaderboard displays correct information once level 3 has been completed	To check if the game will take in the correct user name input and display it with the correct score	Using arrow keys and keyboard to complete all 3 levels and enter name	Valid - Using keyboard	Valid - Game should display the inputted name with the score achieved

Sounds Testing

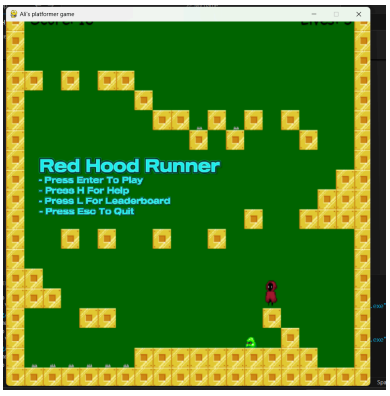
27	Background Music	To check if there is background music while playing the game	N/A	N/A	No expected result as I didn’t have time to add any music to my game

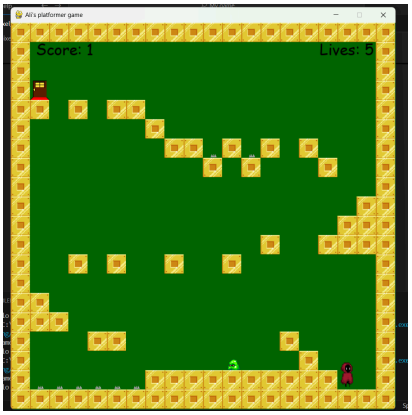
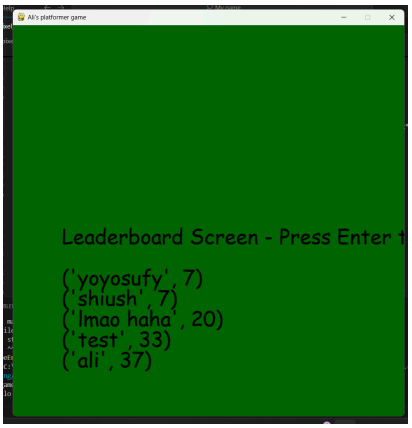
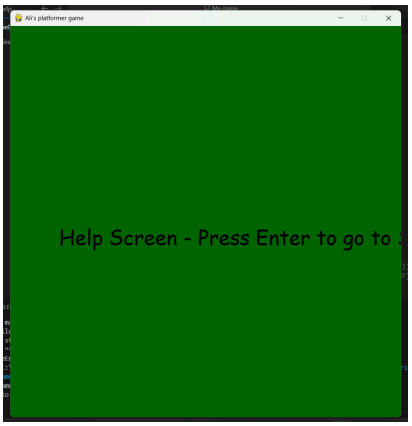
Power Ups Testing

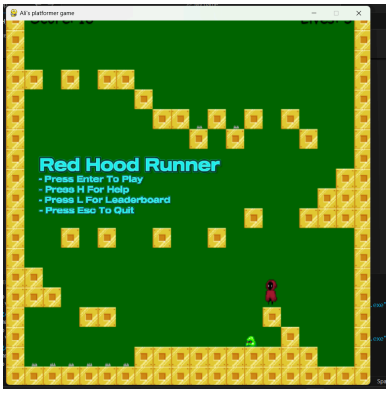
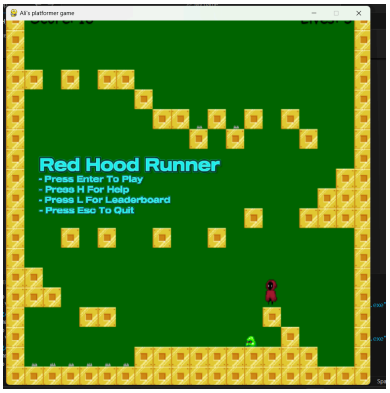
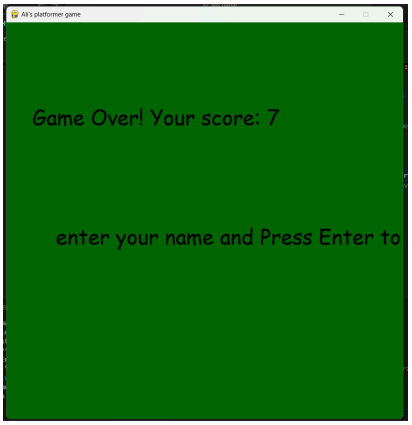
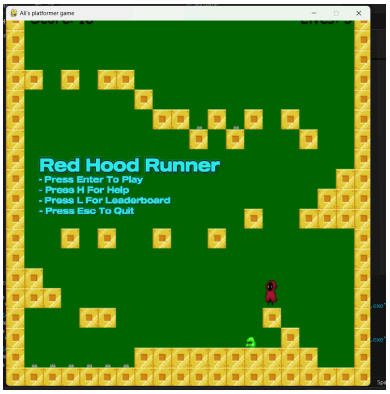
28	Monster chase	If last 3 has a monster which chases you will you try to escape the level	Using arrow keys to navigate the character through the level	Valid - Using arrow keys	Valid - No expected result as requirement was not met as I couldn't implement this into my game
29	Randomly Generated levels	If after level 3 there are any randomly generated	N/A	N/A	No expected results as the system for this would be

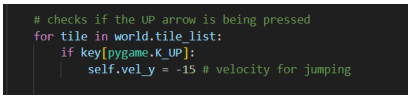
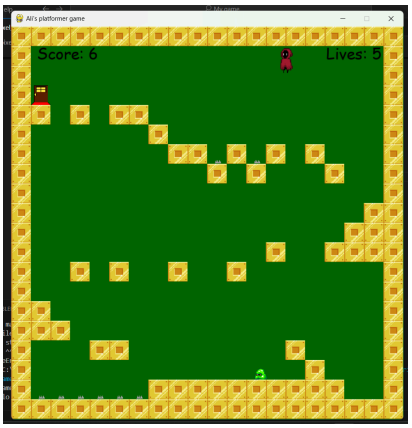
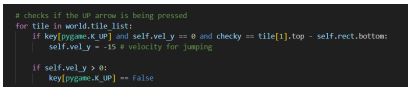
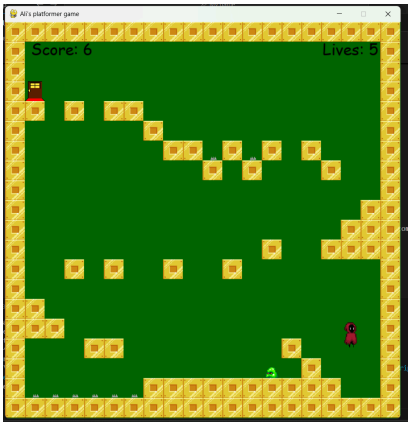
		levels			very complex and I did not have time to even start planning this
30	Different/Custom Characters	If user can customise their character	N/A	N/A	No expected results since I did not get to implement this
31	2 Player Mode	If 2 players can play together with two character on the screen	N/A	N/A	No expected results since I did not implement this
32	Transition Effects	If there are any effects while transitioning between levels	N/A	N/A	No expected results as I did not implement transition effects between levels

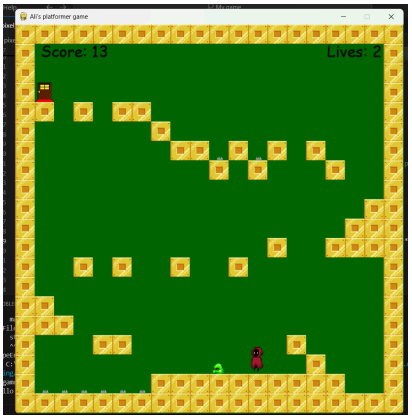
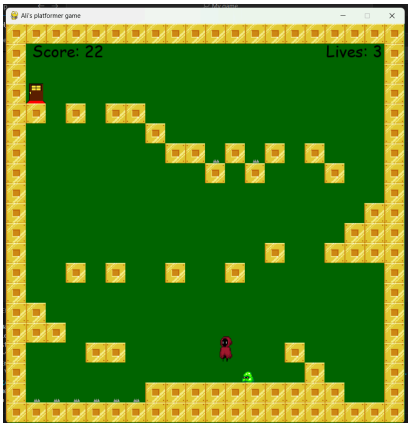
Evidence Table

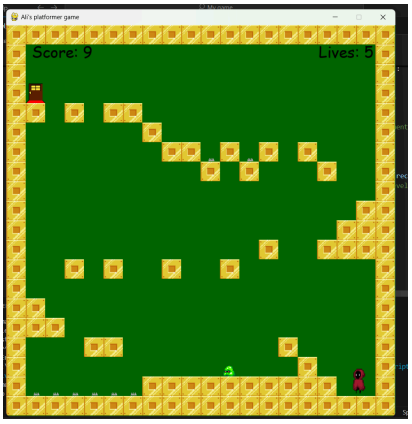
Test No	Evidence	Actual Result
1		Valid - The game loads successfully once start is pressed in visual studio code

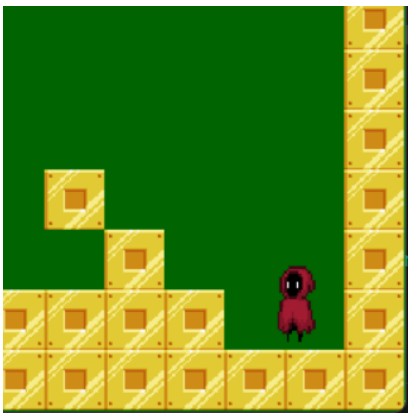
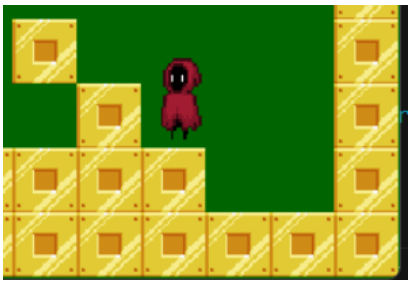
2		Valid - Once ENTER key is pressed level 1 is loaded Invalid - Once T is pressed then nothing happens Analysis - This shows only when the ENTER key is pressed level 1 loads proving that this section works perfectly fine
3		Valid - Once the L key is pressed then the leaderboard screen is shown just as intended Invalid - Once the D key is pressed then nothing happens Analysis - This proves that everything is working perfectly fine exactly how I wanted
4		Valid - Once the H key is pressed then the help screen is loaded as shown here Invalid - Pressing D did nothing which is good as the game shouldn't react to that Analysis - Game works how I want it to can help screen can be loaded
5		Valid - Pressing esc did exit the game leaving me with my window screen Invalid - Pressing shift does nothing Analysis - esc key works in closing my game which is good

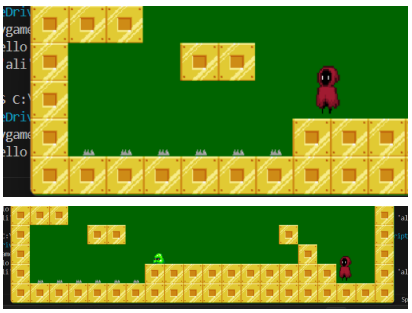
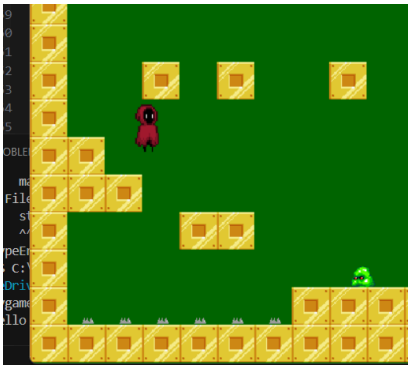
6		Valid - pressing ENTER did move the game back to the main menu Invalid - I tried pressing H to see if the game would go back to the help screen but nothing happened which is what i wanted so the user has to go back to the main menu Analysis - Game works as intended with no problems
7		Valid - pressing ENTER moved the game back to the main menu Invalid - pressing L to see if it would go to the leaderboard screen but nothing happened Analysis - Game went back to the main menu and did nothing when invalid key was pressed meaning it works as intended
8		Valid - Using the arrow keys to move the character into the slime monster five times resulted in the game over screen being shown Analysis - This shows that the code works in identifying that the user has lost all five lives
9		Valid - Pressing the ENTER key when in the game over screen resulted in the main menu screen being shown just as intended Analysis - This shows the game is working exactly as intended since this now allows the user to try again if they want to

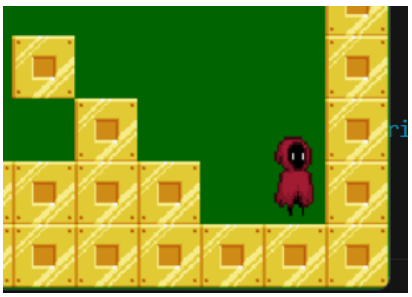
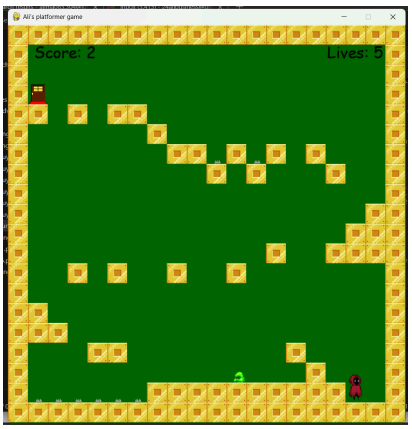
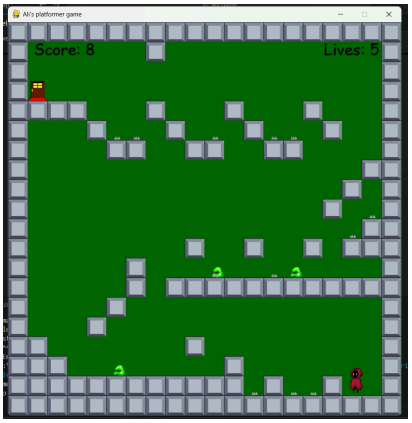
10	   	Valid - Pressing the UP arrow key allowed for my character to jump fie Invalid - pressing U did nothing which is how it's mean to be Problem - the problem I came across was that the user can spam jump to take shortcuts and basically fly away which is not meant to happen Remedial Action - The code for allowing my character to jump had nothing to prevent my character to infinitely jump. Therefore where it checks if the UP arrow key is being pressed I added two extra conditions. These are that the y velocity of my character is zero and that the character is currently colliding with the tile under him. If these are true then the y velocity will increase allowing the character to jump.

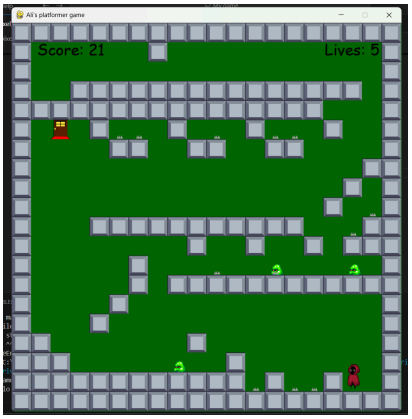
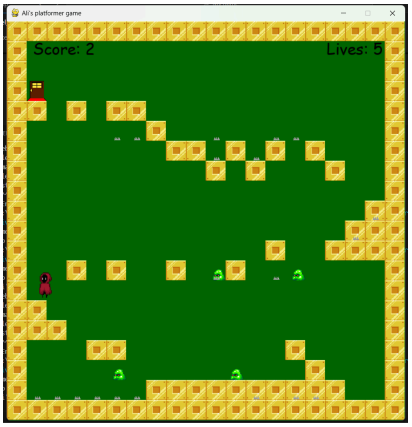
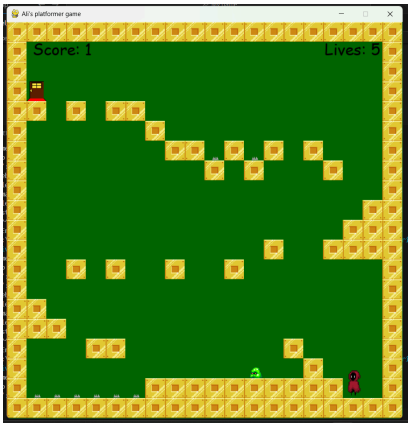
11	
	 
	`self.rect = self.rect.inflate(-20,-20)`
	`# class for enemies`
	`class enemies(pygame.sprite.Sprite):`
	`  def __init__(self, x, y):`
	`  pygame.sprite.Sprite.__init__(self) # initialising`
	`  self.image = pygame.image.load('slime1.png') # load image for slime enemy`
	`  self.rect = self.image.get_rect() # create rectangle for image`
	`  self.rect = self.rect.inflate(-20,-20)`
	`  self.rect.x = x # set x position`
	`  self.rect.y = y # set y position`
	`  self.slime_move = 1 # set speed for slime enemy`
	`  self.slime_counter = 0 # tracks movement distance`
		Valid - it works but collision it's very hard to jump around the slime monster
		
		Problem - collision rectangle of the slime monster is big
		
		Remedial action - to fix the slime monster I will make it smaller and deflate it to allow the character more space to jump around it. I deflated it using the pygame built in feature of inflate and i put in a negative value of -20

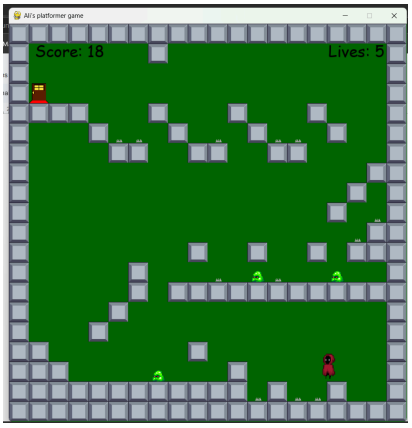
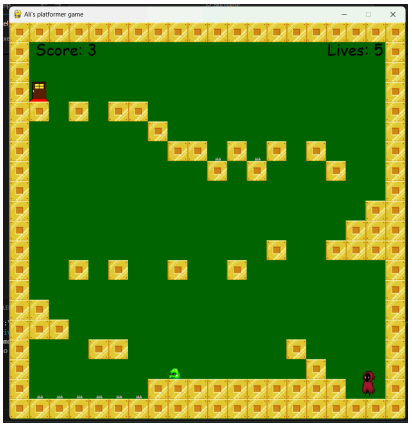
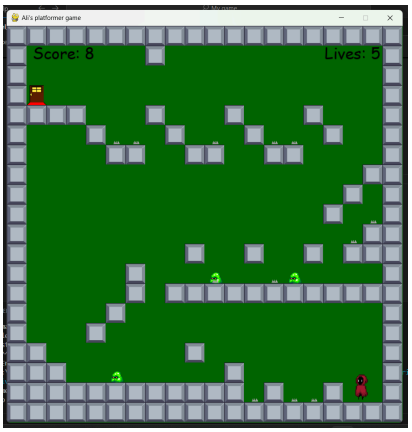
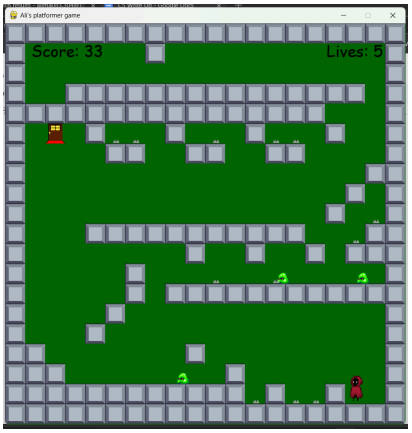
12	 	Valid - The character moves as intended. When the left arrow key is pressed the character moves left then when the right arrow key is pressed the character moves right Invalid - Pressing the T key did nothing which is good meaning the game is doing what its meant to do and is not affected by random key presses Analysis - This shows that the game is working as intended and the user will be able to move horizontally throughout the levels without a problem
13		Valid - Pressing ENTER transitioned the game into level 1 and the character did spawn in the bottom right of the level Invalid - Pressing T did nothing as intended Analysis - The character spawns in which is what is meant to happen so that the user can move them to ge to the transition door

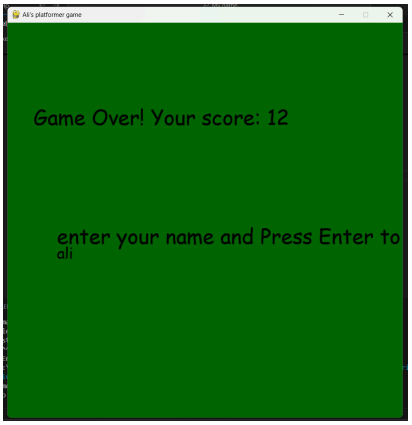
14		Valid - Level 1 loads in and the character is standing on the golden tiles with no problems and is not falling through which allows the user to move the character through the level Invalid - Nothing happens when the I key is pressed Analysis - This shows that the game collision system is working as intended which in turn allows for the user to complete the game
15	 	Valid - Pressing the UP and LEFT arrow key at the same time allows for the character to jump onto another tile Invalid - Pressing X did nothing Analysis - Character can jump onto another tile without falling through it or getting stuck which shows that the keys are working as intended which will allow the user to navigate the character throughout the levels without a problem
16		Valid - Using the UP, LEFT and RIGHT arrow keys I navigated the character towards the slime monster and collided them together, this resulted in the character respawning back to the initial position which is at the bottom right of the level Analysis - This indicates that my collision system is working and that my respawning system is also working allowing the user to try again if they fail

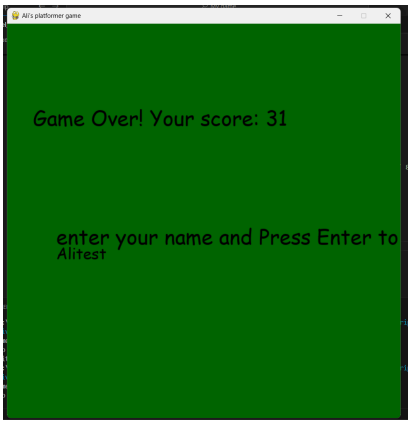
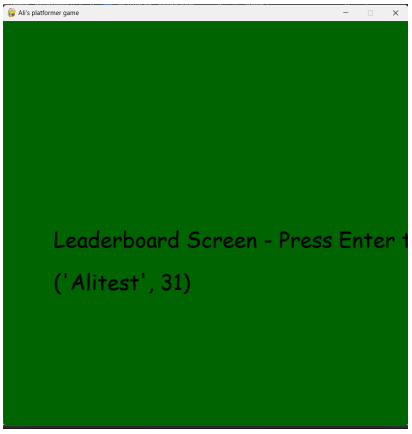
17		Valid - Using the UP, LEFT and RIGHT arrow keys I navigated the character towards the spike trap and collided them together, this resulted in the character respawning back to the initial position which is at the bottom right of the level Analysis - This indicates that my collision system is working and that my respawning system is also working allowing the user to try again if they fail
18		Valid - When pressing the UP arrow key under a tile

19	 	Valid - When using the LEFT and RIGHT arrow keys to move the character into the tiles, character stopped moving not allowing me to move off the screen Analysis - This shows that the collision system works
20		Valid - Once the ENTER key was pressed the game transitioned to level 1 Analysis - This shows that the level system works in loading level one allowing the user to start playing the game
21		Valid - Once the character collided with the transition door the game loaded level 2 and the character spawned in the bottom right of the level Analysis - This shows that the level transitioning system is working and is detecting when level 2 needs to be loaded

22		Valid - Once the character collided with the transition door the game loaded level 3 and the character spawned in the bottom right of the level Analysis - This shows that the level transitioning system is working and is detecting when level 3 needs to be loaded
23	  	Valid - Level one was displayed on the screen however not correctly Problem - Once trying to play the game again, it turned out that the game is loading in the spikes and enemies from level 2 and also it's spawning the character where the transition door of level 3 is which is weird, this also only happens when I complete all 3 levels and try to play again. Also even if I play through it with the extra monsters and enemies once I collide with the transition door the game just freezes while the timer keeps going Remedial Action - In my code I had a part where it checks if the player collides with the transition door specifically on level 3 and so all I did was add to it to empty the groups of enemies and spikes , reset the level number back to 1, reset the player's spawn position and change a variable I have which is called end_game back to zero which indicates the game isn't over

		Analysis - Now when all levels have been completed and the user wants to play again, they can without the game freezing, the character spawning in the wrong position or unwanted spikes and enemies from different levels
24	  	Valid - Once level 1 was loaded the timer started in the top left of the screen Analysis - This shows that the user can easily see what his time/score is whenever they want throughout all 3 levels and that the timer is working just as intended

25	
	`level_3_complete = False`
	`global world, level_number, level_3_complete`
	`def game_over_screen(score):`
	`global level_3_complete`
	`if level_3_complete:`
	`writeScore(name, score) # passes to the function the name and score`
	`level_3_complete = False`
		Valid - After losing all the 5 lives within level 1 the game does ask for my name input which it shouldn't, it should only ask if I've completed level 3
		
		Problem - Name input and score being passed to the leaderboard should only happen once level 3 is completed however it is being done in level 1
		
		Remedial Action - To first fix the score being passed i create a new global variable called level_3_complete, this will be a flag to check when level 3 is completed. Once the character collides with the transition door in level 3 this flag will be set to true. In the game over screen where the score is passed I will only allow that to happen if that flag is set to true preventing it from passing the score in level 1 or 2. I then set the flag back to False so that the process can repeat for the next user
		
		Remedial Action 2 - To then fix the name input showing after losing all lives within level 1 and 2, I went to the game over screen and looked for where draws all the text currently being shown. Before it was always showing the score and asking for name input, however now it only asks for name input once level has been completed. To do this I used the variable that I created earlier which checks when the character collides with the transition door in level 3 and I made it so that it only asks for name input if that variable is set to true.
		
		Analysis - Before all the changes to the code it wasn't

		working as intended since anyone could get their name on the leaderboard. However now only if you complete level 3 will it ask for your name and take in your score to the leaderboard
26	 	Valid - Game does ask for my name once I completed level 3 and then displays my name with the score that I achieved in the leaderboard screen Analysis - This shows that users who complete all 3 levels will have their name displayed on the screen with the score they achieved

Beta Testing

Question No	Question	Answer (Paraphrased)	Analysis
1	Can you pass through each level in a manner where it is not too hard nor too easy ?	Emir - “ Level one is fine and I got through it in a good amount of time, however when I reached level two it was a bit hard and it took longer than I expected. Level three was fine. Overall it was fun and I enjoyed passing through each level ”	After listening to the different responses from my stakeholders I have learnt that an experienced gamer such as Emir didn’t have a problem however Murtadha and Hussain

		Murtadha - “ Level one was okay and it was fun since there was a monster and spikes which were easy to avoid, however level 2 and level 3 were much more harder because there isn’t much space to jump around and avoid the spikes “	don’t play as much as him and found it more difficult to navigate through the levels. After thinking I’ve come to the conclusion that I might change the levels to make them easier to play
		Hussain - “ All the level were hard and it took me a lot of tries to reach level 2, after level 2 I could not pass it and move to level 3 “	
2	Is the help screen and leaderboard screen easily accessible and does the main menu help with accessing them ?	Emir - “ The main menu is helpful in telling me what I need to press if I want to go to the help screen or to the leaderboard screen and even within those screen it tells me how to go back, When pressing the needed keys the game switches to the actual screens and doesn't take a long time which is good “	After reviewing the responses that my stakeholders have given I can see that the main menu works well in informing the user what they need to do in order to go to the help screen and leaderboard screen which is good and so I don’t need to change anything
		Murtadha - “ I can easily access the help screen and leaderboard screen through what the main menu tells me and move through them quickly “	
		Hussain - “ After reading what is displayed on the main menu I can move to the help screen and leaderboard screen easily by pressing H or L “	
3	Can you easily move the character throughout all the levels ?	Emir - “ Yes I’m able to move the character using the arrow keys which is comfortable and it allows me to easily navigate throughout the map, However I would prefer WASD because I am more used to since I play a lot of First Person Shooter games “	Looking at these comments, I am happy with the fact that it’s easy to move the character throughout the levels and that the stakeholders could use it to dodge the different obstacles
		Murtadha - “ Yes, through using the arrow keys I can easily move the character throughout the level and play the game “	
		Hussain - “ I like the fact that I can use the arrow keys to move the character because I’m used to playing 2d games like pacman and it’s easy to do it	
4	Are the enemies challenging or are they too easy ?	Emir - “ Level one was really fun in the way that there was more than one different obstacle I had to avoid , first was the slime enemy and then the spikes which I had to jump over. Level 2 was a little boring and I didn’t enjoy it much however I liked the way level 3 was designed and it’s layout was	Ater looking through the comments my stakeholders made about whether or not the enemies are too challenging, I’ve come to the conclusion that I

		enjoyable”	might redesign either level 1 or level 2’s layout. This is because my stakeholders did mention that level 1 and 3 were challenging and took some time to complete. In my opinion certain levels do need to be challenging because I don’t want the whole game to be easy, however I also don’t want the game to be too hard.
		Murtadha - “ Level one was fun but I found it pretty hard, especially the first jump where I had to go over the slime enemy which kept moving left or right but after multiple tries I did get over it. Level 2 was easy to get through but then level 3 was also challenging “	
		Hussain - “ I found level 1 and 3 the hardest but I could not complete level 3 because of how hard it was. Level 1 was also hard but I managed to get through it and level 2 was easy and fun to get through “	
5	Are there too many levels ?	Emir - “ No, I actually think that the game could be improved by adding more levels because I finished all the levels pretty quickly. I would suggest that adding two more levels would make the game more enjoyable so that there are 5 levels in total and so that the user gets to play the game for even longer “	Looking through the comments my stakeholders have made I agree with them with the fact that there isn’t too many levels but instead I could add more levels to make the game a little longer
		Murtadha - “ I don’t think there are too many levels but I also don’t think there are too few levels. Three levels was fun and enjoyable but it did finish quicker than I expected and so I think it wouldn’t hurt to add a few more levels and it might even make the game more enjoyable “	
		Hussain - “No there aren’t too many levels. I think there is a good amount of levels which give a good amount of variety. They all show different scenarios of how to avoid the enemies and the spikes and the game is very enjoyable “	
6	Are the colours chosen for each level appropriate ?	Emir - “ Yes I found the different colours for each level a nice addition. It gave each level a unique feel and made the game look good. The colours were appropriate and well chosen “	Looking through these comments from the stakeholders I can see that the implementation of having different coloured tiles and backgrounds was a good idea and had a positive impact on the game
		Murtadha - “ I liked the different colours for each level as it made me excited to go to the next level and each level was something new and different “	
		Hussain - “ Yes the colours are appropriate and is a good idea since it makes the game looks better “	

7	Once you complete all three levels can you input your name into the leaderboard ?	Emir - “ Yes once reaching the final level and completing it the game asked me to input my name and when I did that I went back to the main menu and into the leaderboard and there I saw my name with the score I achieved “	Through reviewing these comments, I can say that the game is functioning as intended since it asks for the user’s name only once they complete all three levels which is what I intended
		Murtadha - “ Yes, when I completed all three levels the game went to a different screen asking for my name and to press enter once inputted “	
		Hussain - “ I couldn't reach the last level and so the game never asked for my name and only showed my score whenever I lost all my lives “	
8	What do you think about the timing/score system	Emir - “ I think it’s good as it pushes the user to complete the level as quickly as possible to get the highest score to get their name on the leaderboard “	Reading these comments have made me think that the scoring system is working in the way that it pushes the user to complete the levels as quickly as they can and avoid all obstacles on the way to the end
		Murtadha - “ I like it since it challenges me when playing and pushes me to complete the game in the lowest amount of time possible “	
		Hussain - “ It makes me want to complete all the levels very quickly so that I can get a high score, however sometimes after failing multiple attempts it can be annoying to look at as it’s always there going up which can make you feel bad “	
9	Do you have any suggestions for me ?	Emir - “Overall the game is good, however I feel like it ended really quickly and so I would suggest adding a few more levels as I think it would have a great impact on the game. This would extend the amount of time the user gets to play and let them enjoy the game for longer “	After looking through these comments, I’ve thought about what I could add and I liked the ideas of adding more levels as it was a recurring theme that the game ended pretty quickly and the fact that there aren’t enough lives or a way to earn lives that were lost. I could make it so that if you lose a life but stay alive for a certain amount of time after you regain one life back all the way up to the maximum amount of lives, however then the user would just be able to
		Murtadha - “ I liked the game and enjoyed it but on the way to getting to the last level I failed multiple times and had to restart from level one which was annoying and so I would suggest adding something to help regain lives like health regeneration so that the user can earn back their lives “	
		Hussain - “ I would add two player, like add another character in the game that spawns next to the main character and then so two people can play together through the game which would make it more fun “	

			not do anything and gain back their lives and so I would to add something in place to stop them from doing that. Adding more levels may not be that hard and something that I could do which will improve the game.

Beta Testing - Action Taken

Adding More Levels

- I have decided to add two more levels to my game as I find that this will greatly increase the enjoyment of the game and so that the game will not take as long. Therefore in total the game will have five levels for the user to complete and go through

```
if level_number == 3:
    current_level = level_3 # load level 3 data
    world = levels(current_level) # creates world for third level
    player.rect.x, player.rect.y = player_start_x, player_start_y # sets player's spawn point
    end_game = 0 # changes the state

if level_number == 4:
    current_level = level_4 # load level 4 data
    world = levels(current_level) # creates world for third level
    player.rect.x, player.rect.y = player_start_x, player_start_y # sets player's spawn point
    end_game = 0 # changes the state

if level_number == 5:
    current_level = level_5 # load level 5 data
    world = levels(current_level) # creates world for third level
    player.rect.x, player.rect.y = player_start_x, player_start_y # sets player's spawn point
    end_game = 0 # changes the state
```

- I already have the design and implementation of earlier levels within the game and so it was pretty easy adding more levels. First I started by adding level 4 and 5 to the level handler which will load in it's data, load in the character and set the game status to zero.

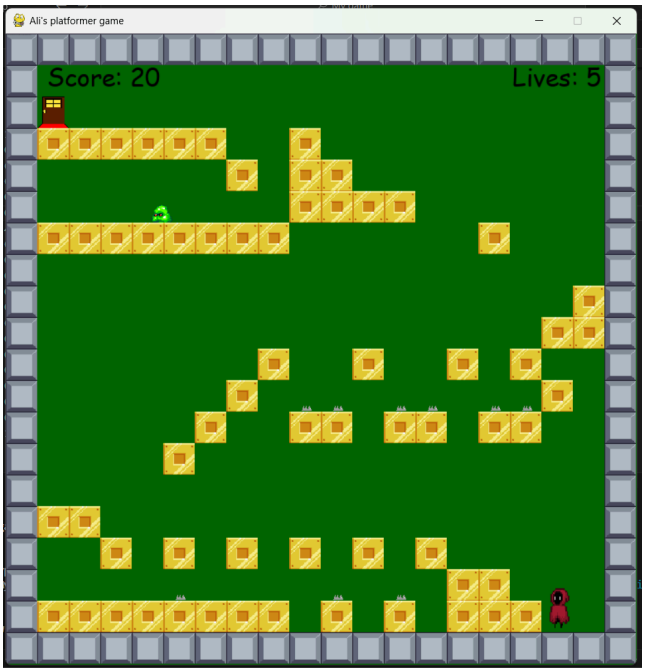
```
if world.transition_block and world.transition_block[1].colliderect(player.rect) and current_level == level_5:
    level_5_complete = True
    slimeguy_group.empty()  # clear enemy group
    spike_group.empty() # clear spike group
    player.rect.x, player.rect.y = player_start_x, player_start_y # sets player's spawn point
    end_game = 0
    level_number = 1
    return GAME_OVER, score
```

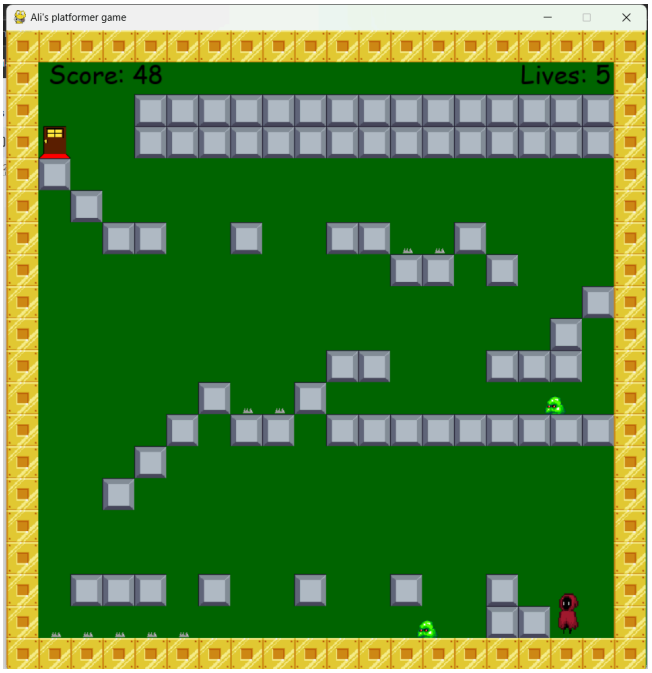
- Since before I had only three levels I had a variable which I used so that when the character collides with the transition door specifically on level 3, the game ends. Now I've changed that variable to level 5 since that's the new last level. As can be seen above level_5_complete is the variable now, I've also changed it where it's needed in other parts of the code

```
# level 4 grid layout
level_4 = [
        [6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,1,1,1,1,1,1,0,0,1,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,0,0,0,0,1,0,1,1,0,0,0,0,0,0,0,6],
        [6,0,0,0,3,0,0,0,0,1,1,1,1,0,0,0,0,0,0,6],
        [6,1,1,1,1,1,1,1,1,0,0,0,0,0,1,0,0,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,6],
        [6,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,6],
        [6,0,0,0,0,0,0,0,1,0,4,0,4,0,4,0,4,1,0,6],
        [6,0,0,0,0,0,0,1,0,0,1,1,0,1,1,0,1,1,0,0,6],
        [6,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,1,0,1,0,1,0,1,0,1,0,1,0,0,0,0,0,6],
        [6,0,0,0,0,0,4,0,0,0,0,0,4,0,4,0,1,1,0,0,0,6],
        [6,1,1,1,1,1,1,1,1,0,1,0,1,0,1,1,1,0,0,6],
        [6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6],
    ]
```

```
# level 5 grid layout
level_5 = [
        [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,1],
        [1,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,6,6,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,6,0,0,6,0,0,6,6,4,4,6,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,6,6,0,6,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,0,1],
        [1,0,0,0,0,0,0,0,0,0,6,6,0,0,0,6,6,6,0,1],
        [1,0,0,0,0,0,6,4,4,6,0,0,0,0,3,0,0,0,0,1],
        [1,0,0,0,0,6,0,6,6,0,6,6,6,6,6,6,6,6,1],
        [1,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,6,6,6,0,6,0,0,6,0,0,6,0,0,6,0,0,1],
        [1,4,4,4,4,4,0,0,0,0,3,0,0,0,0,6,6,0,0,1],
        [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
    ]
```

- After a lot of trial and error I finally got level 4 and 5 designed and are now ready to play





- As can be seen above the new levels are now ready and playable
- Reviewing this remedial action I can say that I have completed it and my game works as intended and now the users who play my game will get to enjoy it for longer

Adding Health Regeneration System

- I have also decided that if I add more levels then it will be much harder for the user to complete them with the same amount of lives. Therefore I am adding a way for the user to gain more lives. I am planning to do this by adding another tile with a red colour which symbolises a heart so that when the character collides with it either by standing on it touching they gain one extra life

```
if tile == 7: # This checks if a 7 is present (gold tile)
    img = pygame.image.load('R.png') # load the image
    img = pygame.transform.scale(img, (tile_size, tile_size)) # resize image
    img_rect = img.get_rect() # create rectangle for image
    img_rect.x = col_count * tile_size # set x position
    img_rect.y = row_count * tile_size # set y position
    self.heart_block = (img, img_rect) # store the heart block
```

- First within my level's class I create a new if statement which will check for a number 7 in the level data grid. I load in in the image, resize it and create a rectangle for it, set the x and y position and assign it to the variable heart_block

```
# draw transition door if needed
if self.heart_block:
    window.blit(self.heart_block[0], self.heart_block[1])
```

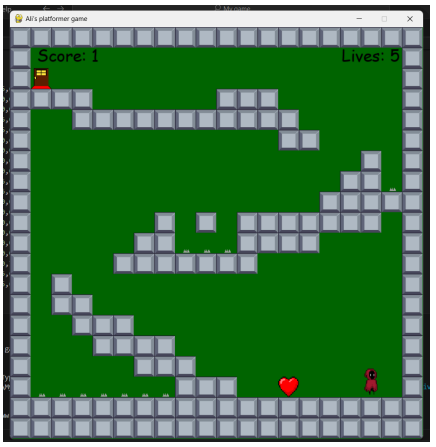
- I then draw it to the screen when needed

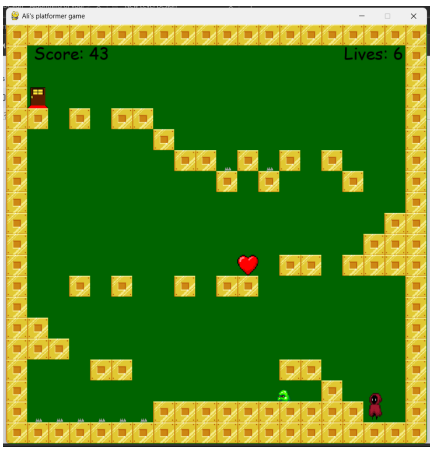
```
self.heart_collected = False  # Track if the heart has been collected
```

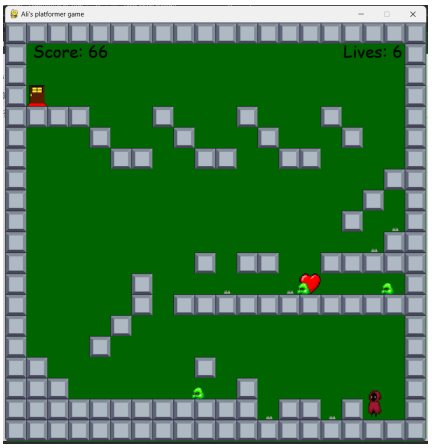
```
if world.heart_block and not world.heart_collected:
    if world.heart_block and world.heart_block[1].colliderect(player.rect):
        lives += 1
        world.heart_collected = True  # Mark heart as collected
        world.heart_block = None  # Remove heart block from the level
```

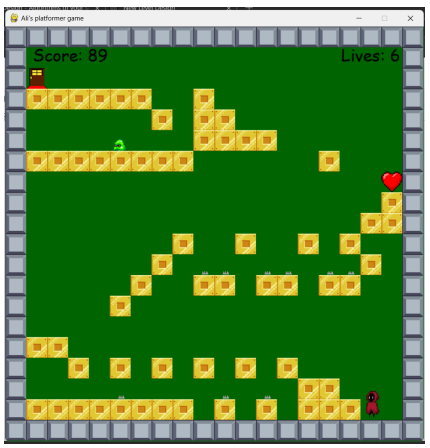
- Here in the playing screen I create new if statements where I use a new variable I created which checks if the character collided with the heart and if they have I increase the lives by one and change it so that the collected variable becomes true and then I remove the heart from the level so that the user cannot keep using it and cheat and get as many lives at they want

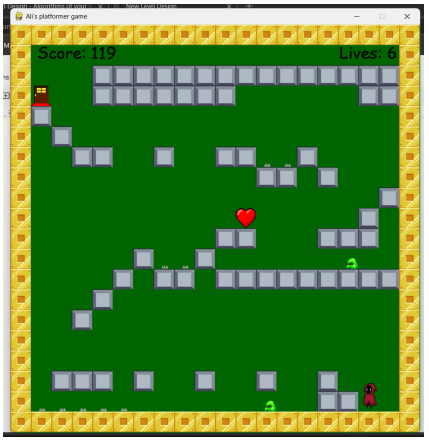
```
# level 3 grid layout
level_1 = [
        [6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,6,6,6,0,0,0,0,0,0,6,6,6,0,0,0,0,0,0,6],
        [6,0,0,6,6,6,6,6,6,6,6,6,6,6,0,0,0,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,6,6,0,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,0,6,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,6,4,6,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,6,6,6,6],
        [6,0,0,0,0,0,0,6,0,6,0,6,6,6,6,6,6,0,6,6],
        [6,0,0,0,0,0,6,6,4,4,4,6,6,6,6,0,0,0,0,6],
        [6,0,0,0,0,6,6,6,6,6,6,6,0,0,0,0,0,0,0,6],
        [6,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,6,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,6,6,6,6,6,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,0,0,6,6,6,0,0,0,0,0,0,0,0,0,0,6],
        [6,4,4,4,4,4,4,4,6,6,6,0,0,7,0,0,0,0,0,6],
        [6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6],
        [6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6],
    ]
```











- Next what I did was add it to each of my level's data grid in a suitable place and now they can be seen on the levels
- Reviewing this action step I am pleased with the outcome now, especially since I added more levels as this means that the game will be harder and the user's will be more likely to lose more lives and so this will now help them

Adding Different Backgrounds For Each Level

- I also thought that the game would look much better and vibrant with better backgrounds for each level instead of the single plain colour which I have right now. So I'm gonna go for pixel themed backgrounds as I think the matches how my game looks like

```
background = pygame.image.load('lvl1.webp') # loading image
background = pygame.transform.scale(background, (WIDTH, HEIGHT)) # transforming to preferre size
```

```
while run: # while run is set to true..
    #window.fill(WHITE) # fills the screen with white colour
    window.blit(background, (0, 0)) # displays the imported background
    world.draw() # draws the tiles
```

Here first I tried loaded the level 1 image in and outputted it and it worked fine however a problem which it caused was the fact that it slowed the game down alot. I think this is because I am loading

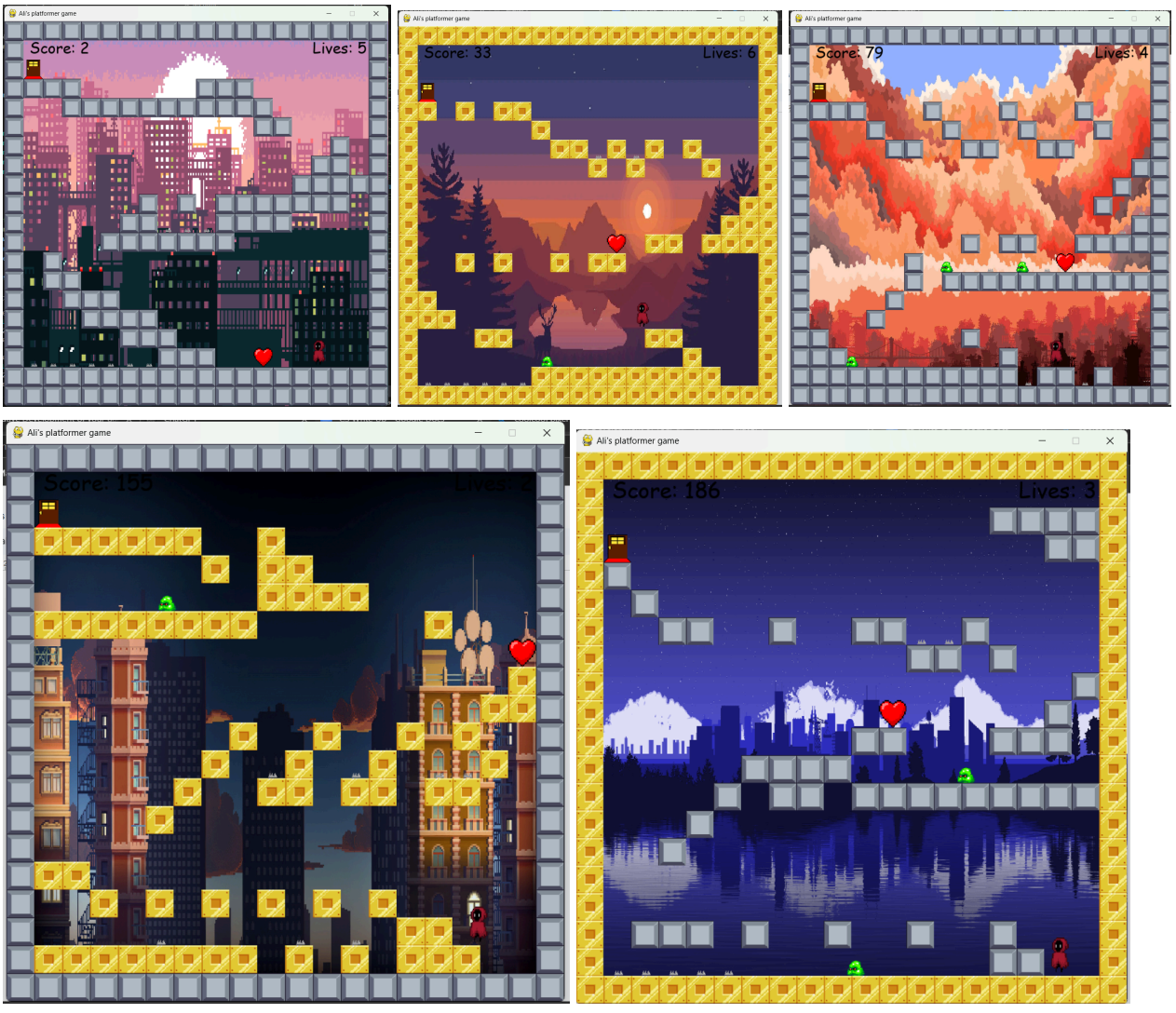
them within the game loop and that means they are being loaded every frame which is not efficient. So after researching and thinking of other ways of doing this, I tried loading them and then storing them in a dictionary and then that way I could select which image I want load it in once at the start of each level

```
backgrounds = {
    1: pygame.transform.scale(pygame.image.load('lvl1.webp'), (WIDTH, HEIGHT)),
    2: pygame.transform.scale(pygame.image.load('extra4.webp'), (WIDTH, HEIGHT)),
    3: pygame.transform.scale(pygame.image.load('extra3.webp'), (WIDTH, HEIGHT)),
    4: pygame.transform.scale(pygame.image.load('extar5.jpg'), (WIDTH, HEIGHT)),
    5: pygame.transform.scale(pygame.image.load('extra1.png'), (WIDTH, HEIGHT))
}
```

- This is the dictionary I created where I loaded each level's background picture in

```
window.blit(backgrounds[level_number], (0, 0))
```

- Here this then outputs and displays the needed image for each level



- As can be seen above, now all the levels have different backgrounds and this makes the game look much better and alive which is good

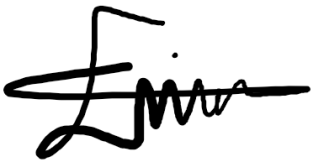
Beta Testing - Stakeholder Responses On Improvement

Emir (paraphrased) - “ After playing the game again there is a big difference as before there was only 3 levels and no way of getting back any lost lives, whereas now I can regain lives back by hitting the heart and the game doesn’t finish as quick since there are two extra levels. I am happy with the improvement of the game. Also the new backgrounds for each level are nice as they make each level unique “

Murtadha (paraphrased) - “ The game feels much better, especially with the new backgrounds for each level as before it felt like a short unfinished game whereas now there are more levels allowing me to enjoy it for longer and it’s easier to stay alive since I can get back lives that I lost helping me to stay alive for longer “

Hussain (paraphrased) - “ I really like this update especially the addition of extra lives because before I used to struggle and I still do, however now I get more attempts to overcome the levels and this was definitely needed because there are more levels. I really do like the way that extra lives have been added through a heart shape placed randomly through the levels as it gives a cool vibe to the game

Signatures: Left - Murtadha, Middle - Emir, Right - Hussain



Evaluation

Testing To Inform

Functionality

Functionality, Robustness, Usability

Video File Name - 7022_H446_Functionality_AliAliMostafa.mp4

Test No.	What Is Being Tested And How	Expected Output	Time Stamp	Pass Or Fail
1	- The game starts without any errors - This is done by pressing start in visual studio code	The game should start and load in the main menu first	00:03 - 00:08	Pass - The game started completely fine with no problem
2	- The different menu's work - This is done by pressing the needed key to transition to the needed screen	The game should display the required screen when each key is pressed	00:20 - 01:00	Pass - The different menus were displayed on the screen when the required key was pressed
3	- The game smoothly transitions into level 1 - This will be done by pressing the ENTER key within the main menu	Level 1 should be loaded and displayed with the character in the bottom right of the screen	01:00 - 01:07	Pass - Level 1 was displayed on the screen when the ENTER key was pressed
4	- The character can be moved left, right and up - This will be done using the arrow keys on the keyboard	The character should moved left, right and then up	01:12 - 01:21	Pass - The character was moved using the arrow keys
5	- The required animations are played when the character moves left, right and up	The required animation are played when the character moves	01:12 - 01:21	Pass - The animation were working when the character was moved

	- This will be tested using the arrow keys on the keyboard			
6	- The lives increase when a heart is collected - This will be tested using the arrow keys on the keyboard	The lives increased by one when the character collected the heart	01:20 - 01:25	Pass - The lives were increased by one when the character collided with the hearts
7	- The character can moves through the level and will die if they touch the slime monster or spike - This will be tested using the arrow keys on the keyboard	The character should move throughout the levels on the tiles and should respawn in the bottom right of the screen when dying	01:25 - 01:37 02:00 - 02:05	Pass - When the character collided with the spikes and the slime monster he respawned in the bottom right of the screen and the lives decreased
8	- The lives mechanism is working - This will be done by colliding with the spikes or monsters using the arrow keys	The lives should decrease by one every time the character dies	01:25 - 01:37	Pass - When the character collided with the spikes he respawned in the bottom right of the screen and the lives decreased
9	- The score mechanism is working	The score should keep increasing by one each second in the top left corner	01:05 - 01:10	Pass - When level 1 was started the score automatically started
10	- When the character collides with the transition door, the level should transition to the next level - This will be tested using the arrow keys on the keyboard	This should cause the game to display the next level or the game over screen if the user is on the last level	01:50 - 01:59	Pass - When the character collided with the transition door in level 1 , level 2 was displayed on the screen
11	- Levels should load without	Each level loads without any errors	01:50 - 01:59	Pass - When the character collided with the transition

	error and glitches - This will be tested using the arrow keys on the keyboard	or delays		door in level 1 , level 2 was displayed on the screen
12	- If on last level the game should transition to the game over screen when completed - This will be tested using the arrow keys on the keyboard	The game should display the game over screen asking for the name of the user	04:32 - 04:38	Pass - when the last level was completed and the character collided with the transition door, the game over screen was displayed
13	- When all levels have been completed and game over screen is displayed , the name that is inputted should be copied to the leaderboard with the score	Leaderboard screen should display the top 5 names with their scores	04:40 - 05:00	Pass - When all 5 levels were completed and the name was inputted , the score and name were copied to the leaderboard which contained them both and sorted them

Robustness

VIdeo FIle Name - 7022_H446_Robustness_AliAliMostafa.mp4

Test No.	What Is Being Tested And How	Expected Output	Time Stamp	Pass Or Fail
1	- WIll game crash if non - used keys are pressed within the main menu, help screen and leaderboard screen - Random keys will be pressed within the main menu, help screen and leaderboard screen	Nothing should happen when random keys are pressed within any of the menus	00:05 - 00:36	Pass - When random keys were pressed nothing happened which is good

2	- Will game crash if random keys are pressed within level 1 - Random keys will be pressed within level 1	Nothing should happen when random non - used keys are pressed	00:36 - 00:45	Pass - Pass - When random keys were pressed nothing happened which is good
3	- Will character stop moving when hitting tiles - This will be tested using the arrow keys on the keyboard	Character should stop moving when colliding with tiles	00:45 - 01:00	Pass - User cannot move the character off the screen as they will be stopped by the tiles
4	- When all lives are lost and user tries to play again are all the level still working normally - This will be tested using the arrow keys on the keyboard	All levels should load fine without error		

Usability - (Uses the functionality video)

VIdeo FIle Name - 7022_H446_Robustness_AliAliMostafa.mp4

Test No.	What Is Being Tested And How	Expected Output	Time Stamp	Pass Or Fail
1	- Can all the different menus be navigated through and understood easily - This will be done using the needed keys	Each menu should be loaded when needed without error	00:20 - 01:00	Pass - The different menus were displayed on the screen when the required key was pressed
2	- Is level 1 loaded fine - This will be done using the ENTER key	Level 1 should load and display the needed tiles and the character fine	01:00 - 01:07	Pass - Level 1 was displayed on the screen when the ENTER key was pressed
3	- After completing all the levels and on the game over	The name inputted should be displayed	04:40 - 05:00	Pass - When all 5 levels were completed and

	screen, can the user easily input their name and then see it on the leaderboard - This will be done using the keyboard	within the leaderboard screen with the score that the user achieved		the name was inputted , the score and name were copied to the leaderboard which contained them both and sorted them

Evaluation Cross Reference

Essential Criteria:

Requirements	Success	Evaluation
Minimum of 3 levels	Fully Met	My game now has 5 levels, before beta testing it was 3 but the stakeholders encouraged me to add more as they said the game felt like it was ending too quickly and so now I think it has a good amount of levels
Main menu with important features	Fully met	The Main Menu displays all the things that the user needs to see such as how to get to the help screen, leaderboard screen and how to quit
Controls for character	Fully Met	The user can use the arrow keys to control the character and they can move Left, Right and Jump up
Timer (score)	Fully Met	There is a score that increases while playing the game, this acts as score mechanism and to win you need to complete all 5 levels in the shortest time possible, this makes the game more competitive
Monsters and Spikes	Fully Met	My game has monsters and spikes which the user needs to dodge and avoid throughout the levels and it gets harder each level to increase the challenge of the game and not keep it boring

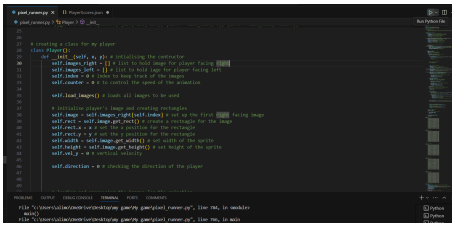
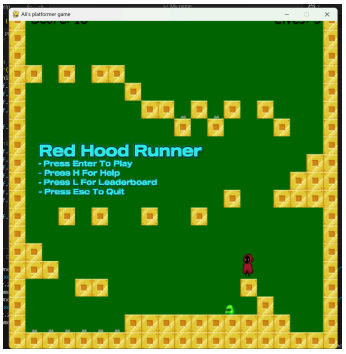
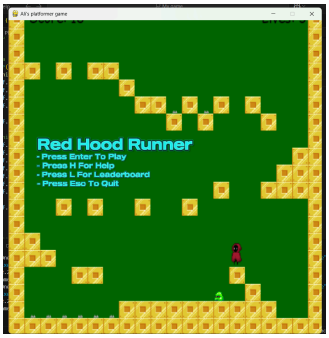
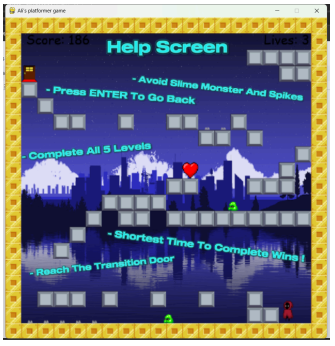
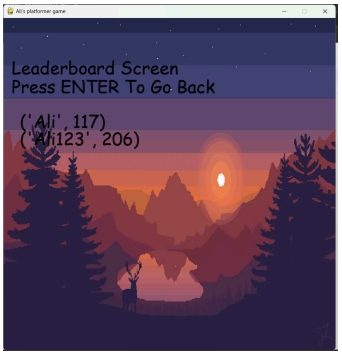
Leaderboard Screen	Fully Met	My game contains a leaderboard screen which allows the user to view who has the highest score, this in turn creates a competitive feel for the game encouraging people to play more and try harder
Help Screen	Fully Met	My game contains a help screen, this informs the user on the main objectives of the game and what to do within each level and how to complete the game
Appeal to audience from children to teenagers	Partially Met	I would say it definitely appeals to children because of the colours and the character within the game but maybe not so much to teenagers since teenagers may find it boring and too basic
Level's progressively get harder	Fully Met	As the user completes each level, the next level is harder to complete. I believe this makes the game more fin because if each level was the same difficulty then the user would get bored pretty quickly
Background Music	Not Met	
Let each level have a different Theme	Fully Met	Each level has a different background with different tile colours alternating between gold and silver tiles and so this makes each level feel unique which makes the game more fun
5 lives for player	Fully Met	My game has 5 lives
Have Bright and vibrant colours	Fully Met	Each of the menus have different colours which are bright and vibrant

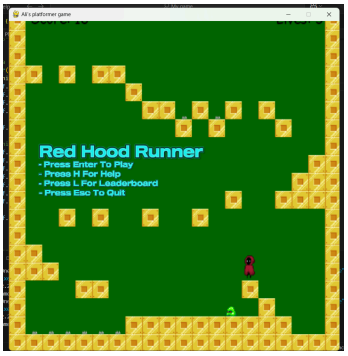
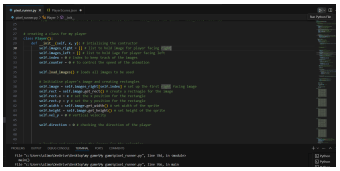
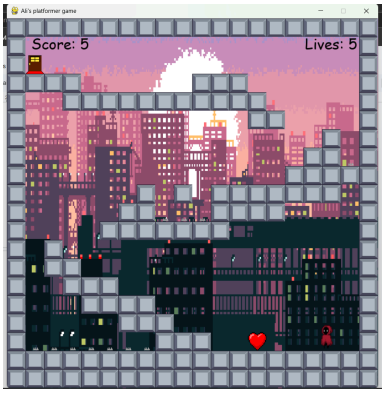
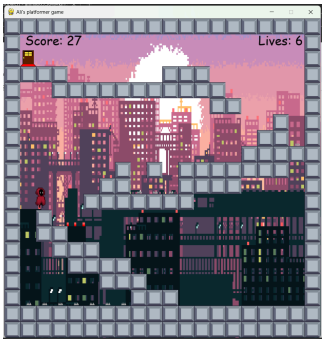
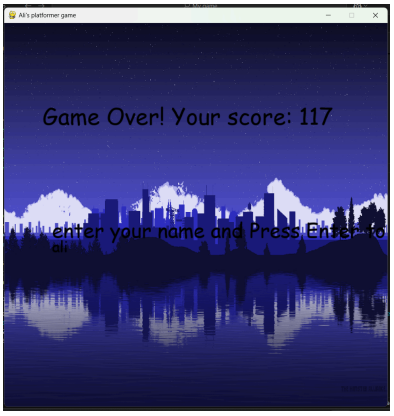
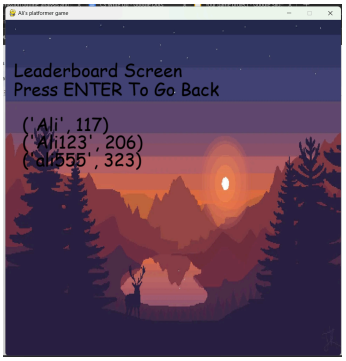
Desirable Criteria:

Requirement	Success	Evaluation
Monster Chase	Not Met	I did not have enough time to add this , but I do believe it would be a good addition
Transition Effects	Not Met	I did not have enough time to

		incorporate this addition
Regain lives mechanism	Fully Met	Within the levels of my game , there are hearts scattered which the user can collect, when collected these hearts increase the amount of

Usability Features

Input	Initial	Final	Justification
Main Menu - Pressing Run In Visual Studio Code			- I would say this was a success since the main menu displays the different things in which the user can do such as going to the help, leaderboard or exiting the game. This is needed because without it the user will not know how to navigate through the game
Help Screen - Pressing H			I would say this was also a success since the Help screen informs the user about the objective of the game and how to get a high position in the leaderboard. It tells the user what to do within the levels and how to go back to the main menu
Leaderboard Screen - Pressing L			This was also a success since the Leaderbaord screen informs the user who has the top 5 highest scores within the game in ascending order. It is in ascending because it's users who complete the game with the

			shortest time who win
Exiting Screen - Pressing esc			Pressing esc allows the user to quit the game without needing to use the mouse to exit it which makes it much easier so this is a success aswell
Moving The Character			The arrow keys are used to move the character within the game. This allows the user to navigate the characters throughout the levels and complete the game so this is also a success
Inputting Name			Once the user completes all 5 levels , the game will ask for their name in the game over screen so that they can put the score and their name together in the leaderboard screen, this was also a success

Success Criteria Overview

My Success Criteria was split up into essential requirements and desirable features. Essential requirements focused on the core parts of the game to make it playable while the desirable features focused on things that aren't really needed but would definitely make the game more fun and look better

- I have fully met almost all of my essential requirements. I had originally planned that the game has three levels but after some feedback from my stakeholders I increased it up to 5. I do believe this was a good addition because the game was ending really quickly but now it takes longer for the user to complete the game and they get to enjoy the different levels for longer. My core game mechanics work well such as the movement for the character, obstacles such as the tiles are displayed fine, enemies and spikes also spawn in with no problems and the scoring and lives mechanisms are working. The score mechanism helps in making the game more competitive which would make users want to play for longer to get the highest score. The user interface works as intended with no problems, the user can enter and exit all the different menus with no problem. Each level has a unique background keeping the game looking good and giving it a good variety of looks and each level progressively get harder to make the game feel more challenging and fun

- Partially met criteria includes whether or not the game appeals to children and teenagers. While I'm sure it will appeal to children, I feel like it may be too simple and childish for older teenagers

Unmet Criteria

- I haven't added in two of my desirable objectives in my success criteria, these include the monster chase and transition effects between levels. The reason for this is due to the complexity of it and also due to time constraints. Adding in a monster chase would make the game much more fun as the last level would be very special and it would also be very fun for the user to have to run away from something. Transition effects would just be more visually appealing than just moving onto the next level. I also don't have any sound, I would think this should be an easy addition however the main reason I haven't added this in is due to time constraints.

  For the Monster chase I would have to find a new sprite to use as the big final monster. I would then add them into my code and use the same settings for them as the normal character. The new code I would have to add is to find a way for them to chase the character throughout the final level and make it spawn a few seconds after the character has started moving to make it fair, I would also need to make it an appropriate speed so that it's not too hard for the user to escape from them.

Maintenance

Current bugs in my game that would need to be fixed if it were to be released

- Firstly, Whenever a user loses all their lives within the level 1 - 4 and then try to restart by playing again, the level in which they died on gets skipped and after level 1 they get spawned in the next level after the one in which they died in. For example, if I were to play and lose all my lives in level 2 and then try to play again, the game would work normally up to level 1 then after completing level I would get spawned in level 3 and so the level in which I died in which is level 2 has been skipped. I have tried fixing this issue but I couldn't resolve it and I believe it is due to the way I've structured my code for my game, I have used multiple loops which made it easier for me to focus on different parts of my game such as the

different menu' s when first learning Python and creating my game, however it's now harder to fix problems such as these.

To fix this, I would look into going very slowly line by line in my code and try to debug it and find the problem. If that wouldn't work then I would look into restructuring my entire game code into a single loop and even though that sounds like it would take a lot of time it will be very much needed.

- Secondly, I do not have pixel perfect collision. When I've tried playing my game, sometimes I get away with slightly skimming the slime monster which is something I do not like as it feels cheap for a game that would be released.

  To fix this I would also add it in a update, I would use a better method of detecting collision than rectangles

Conclusion

- Overall, my game successfully meets most of its essential criteria, it gives the user a playable, engaging experience. The core mechanics, including movement, scoring, lives, and obstacles, function as intended, and feedback from stakeholders allowed me to add valuable improvements, such as extending the game from three to five levels. The user interface is intuitive, and the leaderboard encourages competitiveness among players.
- However, there are areas for improvement. Some desirable features, such as a monster chase and transition effects, were not implemented due to time constraints and complexity. Also, certain bugs, such as the level-skipping issue when restarting, still need fixing. The current structure of my code, using multiple loops, made debugging more difficult, and restructuring it into a single loop would enhance maintainability.
- Despite these limitations, this project has been a valuable learning experience. I have gained a deeper understanding of game development, problem-solving, and the importance of structuring code efficiently. Moving forward, I would focus on refining collision detection, optimising game logic, and incorporating visual and audio enhancements to create a more polished final product. If given more time, I would also work on implementing the missing features to further improve the game's overall appeal and engagement.

Links :

Sources:

- Pixel Adventure by Pixel Frog (itch.io)
  I used this for the tiles in my game
- Hooded Protagonist Animated Character by Penzilla (itch.io)
  I used this for my main character

- SunnyLand Chibi Enemies Pack 1 by ansimuz (itch.io)
  I used this for my slime monster

  Backgrounds I used for my game
- Aesthetic Pixel Art Background HD Wallpapers 49272 - Baltana
- Steam Workshop::Pixel city night v2
- Clean PC, HD wallpaper | Peakpx
-
- Aesthetic Pixel Art Background HD Wallpapers 49272 - Baltana

Appendix :

```python
# Importing Libraries
import pygame
import sys
import os
import json

pygame.init()

# Setting basic structure for game
WIDTH, HEIGHT = 761, 762
FPS = 60
WHITE = (0, 100, 0)
BLACK = (0, 0, 0)
tile_size = 38 # setting the tile size
end_game = 0

# Setting the width, height and name of the game
window = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Ali's platformer game")


# creating a class for my player
class Player():
    def __init__(self, x, y): # intialising the contructor
        self.images_right = [] # list to hold image for player facing right
        self.images_left = [] # list to hold iage for player facing left
        self.index = 0 # Index to keep track of the images
        self.counter = 0 # to control the speed of the animation

        self.load_images() # loads all images to be used

        # Initialise player's image and creating rectangles
        self.image = self.images_right[self.index] # set up the first right facing image
        self.rect = self.image.get_rect() # create a rectnagle for the image
        self.rect.x = x # set the x position for the rectangle
        self.rect.y = y # set the y position for the rectangle
        self.width = self.image.get_width() # set width of the sprite
        self.height = self.image.get_height() # set height of the sprite
        self.vel_y = 0 # vertical velocity

        self.direction = 0 # checking the direction of the player
```

```
41          self.direction = 0 # checking the direction of the player
42
43          # loading and prcoessing the images for the animation
44      def load_images(self):
45          for num in range (1,6):
46              img_right = pygame.image.load(f'hoodguy{num}.png') #load the image facing right
47              img_right = pygame.transform.scale(img_right, (33, 57)) # transform size of image
48              img_left = pygame.transform.flip(img_right, True, False) # flip image to get facing left
49              self.images_right.append(img_right) # add image to list
50              self.images_left.append(img_left) # add image to list
51
52      # Method to check movement
53      def update(self, end_game):
54          checkx = 0 # to track horizontal movement
55          checky = 0 # to track vertival movement
56          sprite_handler = 7 # controls animaton speed
57
58          if end_game == 0: # checks if game is over
59              key = pygame.key.get_pressed() # checks all the keys
60
61              # checks if the UP arrow is being pressed
62              for tile in world.tile_list:
63                  if key[pygame.K_UP] and self.vel_y == 0 and checky == tile[1].top - self.rect.bottom:
64                      self.vel_y = -15 # velocity for jumping
65
66
67              # checks if left key is being pressed
68              if key[pygame.K_LEFT]:
69                  checkx -= 2 # moves player to left by this velocity
70                  self.counter += 1 # increses the counter the control the animation
71                  self.direction = -1 # direction changes to -1
72
73              # checks if right key is being pressed
74              if key[pygame.K_RIGHT]:
75                  checkx += 2 #] moves the player to the right by this velocity
76                  self.counter += 1  # increases the counter to control the animation
77                  self.direction = 1 # direction changes to 1
78
79              # checks if none of the left or right arrow keys are being pressed
80              if key[pygame.K_LEFT] == False and key[pygame.K_RIGHT] == False:
81                  self.counter = 0 # does not change the animation
82                  self.index = 0 # keeps the animation to the first image
83
```

```
80              if key[pygame.K_LEFT] == False and key[pygame.K_RIGHT] == False:
81                  self.counter = 0 # does not change the animation
82                  self.index = 0 # keeps the animation to the first image
83
84                  # selects the needed image based on which direction the player is facing
85                  if self.direction == 1:
86                      self.image = self.images_right[self.index]
87                  if self.direction == -1:
88                      self.image = self.images_left[self.index]
89
90              # handling the animation
91              if self.counter > sprite_handler:
92                  self.counter = 0 # rests to control the animation speed
93                  self.index += 1 # moves to next image for animation
94
95                  # loops back to the first image if index goes to far
96                  if self.index >= len(self.images_right):
97                      self.index = 0
98
99                  # updates the player image based on the direction they are facing
100                 if self.direction == 1:
101                     self.image = self.images_right[self.index]
102                 if self.direction == -1:
103                     self.image = self.images_left[self.index]
104
105             # applying gravity
106             checky += self.vel_y # changes position based on velocity
107             self.vel_y += 1 # slowly increases velocity to make gravity realistic
108
109             # limiting the speed so player does not fall too fast
110             if self.vel_y > 10:
111                 self.vel_y = 10
112
113
114             # collision Checker
115             for tile in world.tile_list:
116                 # x - axis collision
117                 if tile[1].colliderect(self.rect.x + checkx, self.rect.y, self.width, self.height):
118                     checkx = 0 # stops horizontal movement if there is collision
119
120                 # y - axis collision
121                 if tile[1].colliderect(self.rect.x, self.rect.y + checky, self.width, self.height):
122                     if self.vel_y < 0:
```

```
            # y - axis collision
            if tile[1].colliderect(self.rect.x, self.rect.y + checky, self.width, self.height):
                if self.vel_y < 0:
                    checky = tile[1].bottom - self.rect.top
                    self.vel_y = 0 # if player's head aligns with tile's bottom, stop the jump
                elif self.vel_y >= 0:
                    checky = tile[1].top - self.rect.bottom
                    self.vel_y = 0 # if player's bottom aligns with tile's top, stop the movement



        # checks if level finished
        if world.transition_block and world.transition_block[1].colliderect(self.rect.x, self.rect.y, self.width, self.height):
            return "next_level" # if player collides with the transition door, move to next level




        # checks for collision with slime enemy
        if pygame.sprite.spritecollide(self, slimeguy_group, False):
            end_game = -1 # if collision is detected, one live is lost

        # checks collision with spike trap
        if pygame.sprite.spritecollide(self, spike_group, False):
            end_game = -1 # if collision detected, one live is lost



    # updates the player's position
    self.rect.x += checkx
    self.rect.y += checky

    # checks if player has fallenn below screen
    if self.rect.bottom > HEIGHT:
        self.rect.bottom = HEIGHT
        checky = 0


    # draws player's image to the window
    window.blit(self.image, self.rect)
    return end_game
```

```
        return end_game

#level 1 grid layout
level_2 = [
        [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,1,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,1,1,4,1,4,1,0,1,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,1,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,7,0,1,1,0,1,1,1,1],
        [1,0,0,1,0,1,0,0,1,0,1,1,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,1,1,0,0,0,0,0,0,1,1,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,3,0,0,0,0,1,0,0,0,1],
        [1,4,4,4,4,4,4,4,1,1,1,1,1,1,1,1,1,0,0,1],
        [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
    ]

# level 2 grid layout
level_3 = [
        [6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,6,6,6,0,0,0,6,0,0,0,6,0,0,0,6,0,0,0,0,6],
        [6,0,0,0,6,0,0,0,6,0,0,0,6,0,0,0,6,0,0,6],
        [6,0,0,0,0,6,6,0,0,6,6,0,0,6,6,0,0,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,0,4,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,6,6],
        [6,0,0,0,0,0,0,0,0,6,0,6,6,0,0,6,6,6,6,6],
        [6,0,0,0,0,0,0,6,0,0,0,4,3,0,4,7,3,0,0,0,6],
        [6,0,0,0,0,0,6,0,6,6,6,6,6,6,6,6,6,6,6],
        [6,0,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,6,0,0,0,0,0,0,0,6,0,0,0,0,0,0,0,0,6],
```

```
# level 2 grid layout
level_3 = [
        [6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,6,6,6,0,0,0,6,0,0,0,6,0,0,0,6,0,0,0,6],
        [6,0,0,0,6,0,0,0,6,0,0,0,6,0,0,0,6,0,0,6],
        [6,0,0,0,0,6,6,0,0,6,6,0,0,6,6,0,0,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,0,4,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,6,6],
        [6,0,0,0,0,0,0,0,0,6,0,6,6,0,0,6,6,6,6,6],
        [6,0,0,0,0,0,6,0,0,0,4,3,0,4,7,3,0,0,0,6],
        [6,0,0,0,0,0,6,0,6,6,6,6,6,6,6,6,6,6,6,6],
        [6,0,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,6,0,0,0,0,0,0,0,6,0,0,0,0,0,0,0,0,0,6],
        [6,6,6,0,0,0,3,0,0,0,0,6,0,0,0,0,0,0,0,6],
        [6,6,6,6,6,6,6,6,6,6,6,6,4,6,6,4,6,0,0,6],
        [6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6],
    ]


# level 3 grid layout
level_1 = [
        [6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,6,6,6,0,0,0,0,0,0,6,6,6,0,0,0,0,0,0,6],
        [6,0,0,6,6,6,6,6,6,6,6,6,6,6,0,0,0,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,6,6,0,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,6,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,6,6,6],
        [6,0,0,0,0,0,0,6,0,6,0,6,6,6,6,6,6,6,0,6],
        [6,0,0,0,0,0,6,6,4,4,4,6,6,6,6,0,0,0,0,6],
        [6,0,0,0,0,6,6,6,6,6,6,6,0,0,0,0,0,0,0,6],
        [6,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,6,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,6,6,6,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,6,6,6,6,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,0,0,6,6,6,0,0,0,0,0,0,0,0,0,0,6],
        [6,4,4,4,4,4,4,4,4,6,6,6,0,0,7,0,0,0,0,6],
```

```
# level 3 grid layout
level_1 = [
        [6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,6,6,6,0,0,0,0,0,0,6,6,6,0,0,0,0,0,0,6],
        [6,0,0,6,6,6,6,6,6,6,6,6,6,6,0,0,0,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,6,6,0,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,6,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,6,4,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,6,6,6],
        [6,0,0,0,0,0,0,6,0,6,0,6,6,6,6,6,6,6,0,6],
        [6,0,0,0,0,0,6,6,4,4,4,6,6,6,6,6,0,0,0,6],
        [6,0,0,0,0,6,6,6,6,6,6,6,6,0,0,0,0,0,0,6],
        [6,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,6,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,6,6,6,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,6,6,6,6,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,0,0,6,6,6,0,0,0,0,0,0,0,0,0,0,6],
        [6,4,4,4,4,4,4,4,4,6,6,6,0,0,7,0,0,0,0,6],
        [6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6],
        [6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6],
    ]


# level 4 grid layout
level_4 = [
        [6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,1,1,1,1,1,1,0,0,1,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,3,0,0,0,0,1,1,1,1,0,0,0,0,0,0,6],
        [6,1,1,1,1,1,1,1,1,1,0,0,0,0,0,1,0,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,6],
        [6,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,6],
        [6,0,0,0,0,0,0,1,0,4,1,0,4,1,0,4,4,1,0,6],
        [6,0,0,0,0,0,1,0,0,1,1,0,1,1,0,1,1,0,0,6],
        [6,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,0,0,6],
        [6,0,0,0,0,0,4,0,0,0,0,4,0,0,4,0,1,1,0,0,0,6]
```

```
# level 4 grid layout
level_4 = [
        [6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,1,1,1,1,1,1,0,0,1,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,3,0,0,0,0,1,1,1,0,0,0,0,0,0,0,6],
        [6,1,1,1,1,1,1,1,1,0,0,0,0,0,1,0,0,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,6],
        [6,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,6],
        [6,0,0,0,0,0,0,1,0,4,1,0,4,1,0,4,4,1,0,6],
        [6,0,0,0,0,0,1,0,0,1,1,0,1,1,0,1,1,0,0,6],
        [6,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6],
        [6,0,0,1,0,1,0,1,0,1,0,1,0,1,0,0,0,0,0,6],
        [6,0,0,0,0,4,0,0,0,0,4,0,4,0,1,1,0,0,0,6],
        [6,1,1,1,1,1,1,1,1,0,1,0,1,0,1,1,1,0,0,6],
        [6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6],
    ]


# level 5 grid layout
level_5 = [
        [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,6,6,6,6,6,1],
        [1,5,0,0,0,0,0,0,0,0,0,0,0,0,0,6,6,6,1],
        [1,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,6,6,0,0,6,0,0,6,6,4,4,6,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,6,6,0,6,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,1],
        [1,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,0,6,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,6,6,0,0,0,6,6,6,1],
        [1,0,0,0,0,0,0,6,6,6,6,0,0,0,0,3,0,0,0,1],
        [1,0,0,0,0,6,0,6,6,0,6,6,6,6,6,6,6,6,6,1],
        [1,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,6,6,6,0,6,0,0,6,0,0,6,0,0,6,0,0,0,1],
```

```
# level 5 grid layout
level_5 = [
        [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,6,6,6,1],
        [1,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,6,1],
        [1,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,6,6,0,0,6,0,0,6,6,4,4,6,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,6,6,0,6,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,0,6,1],
        [1,0,0,0,0,0,0,0,0,0,0,6,6,0,0,0,6,6,6,1],
        [1,0,0,0,0,0,0,6,6,6,6,0,0,0,0,3,0,0,0,1],
        [1,0,0,0,0,6,0,6,6,6,0,6,6,6,6,6,6,6,6,1],
        [1,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
        [1,0,6,6,6,0,6,0,0,6,0,0,6,0,0,6,0,0,0,1],
        [1,4,4,4,4,4,4,0,0,0,0,3,0,0,0,0,6,6,0,1],
        [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
    ]
# Function for levels
class levels():
    def __init__(self, data): # Constructor taking in self and data as arguments
        self.tile_list = [] # creating a list
        self.gold_img = pygame.image.load('gold.png') # loading in the gold tile image
        self.silver_img = pygame.image.load('silver.png') # loading in the silver tile image
        self.heart_collected = False  # Track if the heart has been collected
        self.transition_block = None
        self.heart_block = None
        self.grid_select(data)

    def grid_select(self, data):
        row_count = 0 # start from first row and continues
        for row in data:
            col_count = 0 # starts from first column and continues
            for tile in row:

                if tile == 1: # This checks if a 1 is present (gold tile)
                    img = pygame.transform.scale(self.gold_img, (tile_size, tile_size)) # transforms image to preferred size
                    img_rect = img.get_rect() # creates rectangle for image
                    img_rect.x = col_count * tile_size # sets the x position
                    img_rect.y = row_count * tile_size # sets the y position
```

```
302             if tile == 1: # This checks if a 1 is present (gold tile)
303                 img = pygame.transform.scale(self.gold_img, (tile_size, tile_size)) # transforms image to preferred size
304                 img_rect = img.get_rect() # creates rectangle for image
305                 img_rect.x = col_count * tile_size # sets the x position
306                 img_rect.y = row_count * tile_size # sets the y position
307                 tile = (img, img_rect) # creates a tuple for image and rectangle
308                 self.tile_list.append(tile) # adds the tile to the tile list
309 
310             # checks if a 3 is present (slime enemy)
311             if tile == 3:
312                 slimeguy = enemies(col_count * tile_size, row_count * tile_size) # creates enemy object
313                 slimeguy_group.add(slimeguy) # adds to enemy group
314 
315             # checks if a 4 is present (spike trap)
316             if tile == 4:
317                 Spike = spike(col_count * tile_size, row_count * tile_size + (tile_size // 2)) # creates a spike object
318                 spike_group.add(Spike) # adds spike to spike group
319 
320             # checks if a 5 is present(transition door)
321             if tile == 5:
322                 img = pygame.image.load('door.png') # load the image
323                 img = pygame.transform.scale(img, (tile_size, tile_size)) # resize image
324                 img_rect = img.get_rect() # create rectangle for image
325                 img_rect.x = col_count * tile_size # set x position
326                 img_rect.y = row_count * tile_size # set y position
327                 self.transition_block = (img, img_rect) # store the transition door
328 
329             # checks if a 6 is present (silver tile)
330             if tile == 6:
331                 img = pygame.transform.scale(self.silver_img, (tile_size, tile_size)) # resize image
332                 img_rect = img.get_rect() # create a rectangle for image
333                 img_rect.x = col_count * tile_size # set x position
334                 img_rect.y = row_count * tile_size # set y position
335                 tile = (img, img_rect) # create a tuple for image and rectangle
336                 self.tile_list.append(tile) # add tiel to tile list
337 
338             if tile == 7: # This checks if a 7 is present
339                 img = pygame.image.load('R.png') # load the image
340                 img = pygame.transform.scale(img, (tile_size, tile_size)) # resize image
341                 img_rect = img.get_rect() # create rectangle for image
342                 img_rect.x = col_count * tile_size # set x position
343                 img_rect.y = row_count * tile_size # set y position
344                 self.heart_block = (img, img_rect) # store the heart block
```

```
                    img = pygame.transform.scale(img, (tile_size, tile_size)) # resize image
                    img_rect = img.get_rect() # create rectangle for image
                    img_rect.x = col_count * tile_size # set x position
                    img_rect.y = row_count * tile_size # set y position
                    self.heart_block = (img, img_rect) # store the heart block


                col_count += 1 # go to next column
            row_count += 1 # go to next row

    # method which will draw  what is needed
    def draw(self):
        for tile in self.tile_list: # draw all tiles needed
            window.blit(tile[0], tile[1])

        # draw transition door if needed
        if self.transition_block:
            window.blit(self.transition_block[0], self.transition_block[1])

        # draw transition door if needed
        if self.heart_block:
            window.blit(self.heart_block[0], self.heart_block[1])


# class for enemies
class enemies(pygame.sprite.Sprite):
    def __init__(self, x, y):
        pygame.sprite.Sprite.__init__(self) # initialising
        self.image = pygame.image.load('slimer1.png') # load image for slime enemy
        self.rect = self.image.get_rect() # create rectangle for image
        self.rect = self.rect.inflate(-50,-50)
        self.rect.x = x # set x position
        self.rect.y = y # set y position
        self.slime_move = 1 # set speed for slime enemy
        self.slime_counter = 0 # tracks movement distance

    def update(self):
        self.rect.x += self.slime_move # updates the slime's x position
        self.slime_counter += 0.3 # slowly increase the movement
        if abs(self.slime_counter) > 40: # after  a certian distance chnage direction
            self.slime_move *= -1 # move opposite direction
            self.slime_counter *= -1 # chnage the movement counter

```

```
380                 self.slime_move *= -1 # move opposite direction
381                 self.slime_counter *= -1 # chnage the movement counter
382 
383 # class for spike trap
384 class spike(pygame.sprite.Sprite):
385     def __init__(self, x, y):
386         # initialising
387         pygame.sprite.Sprite.__init__(self)
388         image = pygame.image.load('spike.png') # load spike image
389         self.image = pygame.transform.scale(image, (tile_size, tile_size // 2)) # transform image to fit
390 
391         self.rect = self.image.get_rect() # create rectangle for image
392         self.rect.x = x # set x position
393         self.rect.y = y # set y position
394 
395 player = Player(650, HEIGHT - 130) # creating an instance of my player and setting his spawn position
396 spike_group = pygame.sprite.Group() # creating a sprite group for my spikes
397 slimeguy_group = pygame.sprite.Group() # creating a sprite group for my slime enemy
398 world = levels(level_1) # creating an instance by passing in the level layout
399 font = pygame.font.SysFont("comicsans", 50)
400 
401 # Setting constants for each state of the game
402 START = "start"
403 PLAY = "play"
404 GAME_OVER = "game_over"
405 LEVEL_SCREEN = "level_screen"
406 LEADERBOARD = "leaderboard"
407 HELP = "help"
408 
409 #Function to draw text for each state
410 def draw_text(text, size, color, x, y): #Creating a function to draw text onto screen
411     font = pygame.font.SysFont("comicsans", size) #Assigning the font for the text
412     label = font.render(text, True, color) #Render as image
413     window.blit(label, (x, y)) #Blit image to screen
414 
415 # function for main menu
416 def main_menu():
417 
418     background = pygame.image.load('66.png') # loading image
419     background = pygame.transform.scale(background, (WIDTH, HEIGHT)) # transforming to preferre size
420     run = True
421     while run:
422         window.blit(background, (0, 0)) # displays the imported background
```

```
    background = pygame.image.load('66.png') # loading image
    background = pygame.transform.scale(background, (WIDTH, HEIGHT)) # transforming to preferre size
    run = True
    while run:
        window.blit(background, (0, 0)) # displays the imported background
        #window.fill(WHITE)
        #draw_grid()
        #draw_text("Start Screen - Press Enter to Play", 40, BLACK, WIDTH // 4, HEIGHT // 2)
        #draw_text("Press H for Help, Esc to Quit", 30, BLACK, WIDTH // 4, HEIGHT // 2 + 50)
        pygame.display.update()


        # For loop to allow user to navigate through the game
        for event in pygame.event.get(): # Checks for events
            if event.type == pygame.QUIT: # Checks if user wants to quit
                run = False # Closes application
            if event.type == pygame.KEYDOWN: # Checks if a key is pressed
                if event.key == pygame.K_RETURN: #Checks if Enter key is pressed
                    return PLAY # Outputs playing state
                elif event.key == pygame.K_h:  # Checks if H key is pressed
                    return HELP # Outputs help state
                elif event.key == pygame.K_l:  #Checks if L key is pressed
                    return LEADERBOARD # #Outputs Leaderboard state
                elif event.key == pygame.K_ESCAPE:  # Checks if esc key is pressed
                    pygame.quit() # Closes application
                    sys.exit()

level_number = 1 # variable to keep track of levels
#lives = 5

backgrounds = {
    1: pygame.transform.scale(pygame.image.load('lvl1.webp'), (WIDTH, HEIGHT)),
    2: pygame.transform.scale(pygame.image.load('extra4.webp'), (WIDTH, HEIGHT)),
    3: pygame.transform.scale(pygame.image.load('extra3.webp'), (WIDTH, HEIGHT)),
    4: pygame.transform.scale(pygame.image.load('extar5.jpg'), (WIDTH, HEIGHT)),
    5: pygame.transform.scale(pygame.image.load('extra1.png'), (WIDTH, HEIGHT))
}
```

```
458  # function for playing state
459  def play_screen():
460      global world, level_number, level_5_complete # helps accessing the global variable "world" etc
461      run = True
462      end_game = 0 #tracks the game state
463      lives = 5 # number of player lives
464      player_start_x, player_start_y = 650, HEIGHT - 130 # player's starting position
465      score = 0  # score starts form zero
466      score_update_time = pygame.time.get_ticks() # timer for score
467      level_5_complete = False    # **Reset Level Number and Groups if Restarting**
468
469
470
471
472      current_level = level_1 # loads the game data
473      world = levels(current_level) #loads the level
474
475
476      while run: # while run is set to true..
477          #window.fill(WHITE) # fills the screen with white colour
478          window.blit(backgrounds[level_number], (0, 0))
479          world.draw() # draws the tiles
480
481
482          # only updates if game is not closed
483          if end_game == 0:
484              slimeguy_group.update()
485
486          # draws what is needed to game window
487          slimeguy_group.draw(window)
488          spike_group.draw(window)
489
490          # updates player state
491          end_game = player.update(end_game)
492
493          if lives < 1:
494              slimeguy_group.empty()  # clear enemy group
495              spike_group.empty() # clear spike group
496
497          if lives <= 0:  # Assuming you have a flag for completing the Game Over screen
498              # Reset the game state
499              level_number = 1  # Start from level 1
500              score = 0  # Reset score
501              lives = 5  # Reset lives to the starting value
```

```
            # Reset the game state
            level_number = 1  # Start from level 1
            score = 0  # Reset score
            lives = 5  # Reset lives to the starting value
            end_game = 0  # Reset the end_game state
            slimeguy_group.empty()  # Clear all enemy groups
            spike_group.empty()  # Clear all spike groups

            # Reset collected items
            world.heart_collected = False  # Allow the heart to be collectible again

            # Reload the first level
            current_level = level_1  # Load level 1 data
            world = levels(current_level)  # Create the level 1 world
            player.rect.x, player.rect.y = player_start_x, player_start_y  # Reset player spawn point

            # Reset any other state variables related to gameplay
            level_5_complete = False  # If you have a flag for completing level 5


        # checks for level transition
        if end_game == "next_level":
            level_number += 1 # increases by one once "next level" is ouputted
            slimeguy_group.empty()  # clear enemy group
            spike_group.empty() # clear spike group


            if level_number == 2:
                current_level = level_2  # load level 2 data
                world = levels(current_level)  # creates world for second level
                player.rect.x, player.rect.y = player_start_x, player_start_y  # sets player's spawn point for second level
                end_game = 0 # changes the state

            if level_number == 3:
                current_level = level_3 # load level 3 data
                world = levels(current_level) # creates world for third level
                player.rect.x, player.rect.y = player_start_x, player_start_y # sets player's spawn point
                end_game = 0 # changes the state

            if level_number == 4:
                current_level = level_4 # load level 4 data
                world = levels(current_level) # creates world for third level
                player.rect.x, player.rect.y = player_start_x, player_start_y # sets player's spawn point
                end_game = 0 # changes the state

```

```
                current_level = level_4 # load level 4 data
                world = levels(current_level) # creates world for third level
                player.rect.x, player.rect.y = player_start_x, player_start_y # sets player's spawn point
                end_game = 0 # changes the state

            if level_number == 5:
                current_level = level_5 # load level 5 data
                world = levels(current_level) # creates world for third level
                player.rect.x, player.rect.y = player_start_x, player_start_y # sets player's spawn point
                end_game = 0 # changes the state



        if world.transition_block and world.transition_block[1].colliderect(player.rect) and current_level == level_5:
            level_5_complete = True
            slimeguy_group.empty()  # clear enemy group
            spike_group.empty() # clear spike group
            player.rect.x, player.rect.y = player_start_x, player_start_y # sets player's spawn point
            end_game = 0
            level_number = 1
            return GAME_OVER, score


        if world.heart_block and not world.heart_collected:
            if world.heart_block and world.heart_block[1].colliderect(player.rect):
                lives += 1
                world.heart_collected = True  # Mark heart as collected
                world.heart_block = None  # Remove heart block from the level


        # updates score every second
        current_time = pygame.time.get_ticks()
        if current_time - score_update_time > 1000:
            score += 1 # increases score by one each second
            score_update_time = current_time


        # handles lives and game over
        if end_game == -1:  # if player loses
            lives -= 1  # decrease lives by one
            if lives > 0: # if player has not lost all lives
                player.rect.x, player.rect.y = player_start_x, player_start_y # spawn player again
```

```
583             else:
584                 slimeguy_group.empty()  # clear enemy group
585                 spike_group.empty() # clear spike group
586                 player.rect.x, player.rect.y = player_start_x, player_start_y # spawn player again
587                 current_level = 1
588                 return GAME_OVER, score # if no lives are left, go to game over state
589 
590 
591 
592 
593 
594         # displays lives and score onto screen
595         draw_text(f"Lives: {lives}", 30, BLACK, WIDTH - 150, 30)
596         draw_text(f"Score: {score}", 30, BLACK, 50, 30)
597 
598         draw_text("Play Screen - Press L for Level Screen, G for Game Over", 0, BLACK, WIDTH // 8000, HEIGHT // 2000)
599         pygame.display.update() # updates screen
600 
601         # event handler
602         for event in pygame.event.get():
603             if event.type == pygame.QUIT: # checks if user closes window
604                 run = False # end loop
605             if event.type == pygame.KEYDOWN: # checks for any keys pressed
606                 if event.key == pygame.K_l:  # checks if key L is pressed
607                     return LEVEL_SCREEN # go to level screen
608                 elif event.key == pygame.K_g:  # checks if key G is pressed
609                     return GAME_OVER # go to game over state
610                         # Normal return with state and score
611                     return "GAME_OVER", 100  # or whatever the final state and score are
612                     return "EXIT", 0  # fallback values if needed
613 
614 
615 #Used to reset the leaderbaord
616 #empty_data = {}  # Create an empty dictionary to reset the scores
617 #with open('data/PlayerScores.json', 'w') as file:
618 #    json.dump(empty_data, file)
619 
620 
621 # Function to save score to file
622 def WriteScore(name, score): # takes in arguments
623     with open('data/PlayerScores.json', 'r') as file: # opens JSON file read mode
624         data = json.load(file) # loads data of file into dictionary called data
625     data[name] = score # updates the score in data
626     with open('data/PlayerScores.json', 'w') as f: # opens JSON file in write mode
627         json.dump(data, f) # updates file with new data
```

```
621  # Function to save score to file
622  def WriteScore(name, score): # takes in arguments
623      with open('data/PlayerScores.json', 'r') as file: # opens JSON file read mode
624          data = json.load(file) # loads data of file into dictionary called data
625      data[name] = score # updates the score in data
626      with open('data/PlayerScores.json', 'w') as f: # opens JSON file in write mode
627          json.dump(data, f) # updates file with new data
628      print(data) # prints the updated data
629
630  # Creating a function for the game over state
631  def game_over_screen(score):
632      background = pygame.image.load('extra1.png') # loading image
633      background = pygame.transform.scale(background, (WIDTH, HEIGHT)) # transforming to preferre size
634      global level_5_complete, level_number
635      run = True # Sets the state to true
636      name = "" # Store name as an empty string
637      while run: # While run is true, the following will happen...
638          window.blit(background, (0, 0)) # displays the imported background
639
640
641          if level_5_complete:
642              draw_text("enter your name and Press Enter to go to Start Screen", 40, BLACK, WIDTH // 8, HEIGHT // 2) # Displays the text
643              draw_text(name, 30, BLACK, WIDTH // 8, HEIGHT // 2 + 40) # shows written name
644              draw_text(f"Game Over! Your score: {score}", 45, BLACK, WIDTH // 10, HEIGHT // 5)
645          else:
646              draw_text(f"Game Over! Your score: {score}", 45, BLACK, WIDTH // 10, HEIGHT // 5)
647
648
649
650          pygame.display.update() # updates the screen
651
652          for event in pygame.event.get(): # checks for events
653              if event.type == pygame.QUIT:
654                  run = False # If user wants to quit , it will stop running
655              if event.type == pygame.KEYDOWN: # checks if any keys are pressed
656                  if event.key == pygame.K_RETURN: # checks if enter key is pressed
657                      if level_5_complete:
658                          WriteScore(name, score) # passes to the function the name and score
659                          level_number = 1
660                      level_5_complete = False
661                      return START # outputs the main menu
662                  elif event.key == pygame.K_BACKSPACE:
663                      name = name[:-1]  # Remove last character
664                  else:
```

```
                    name = name[:-1]  # Remove last character
                else:
                    name += event.unicode # allows user to write their name


# Function to create leaderboard screen
def leaderboard_screen():
    background = pygame.image.load('extra4.webp') # loading image
    background = pygame.transform.scale(background, (WIDTH, HEIGHT)) # transforming to preferre size
    run = True
    while run: # While run is set to true..
        window.blit(background, (0, 0)) # displays the imported background
        #window.fill(WHITE) #Fills the screen with white colour
        draw_text("Leaderboard Screen", 40, BLACK, WIDTH // 40, HEIGHT // 9) # draws
        draw_text("Press ENTER To Go Back", 40, BLACK, WIDTH // 40, HEIGHT // 6) # draws

        index = 1 # Keeps track of each score entry
        with open('data/PlayerScores.json', 'r') as file: # Opens file in read mode
            data = json.load(file) # Loads file into dictionary
            data = list(data.items()) # changes dictionary into list of tuples
            for mx in range(len(data)-1, -1, -1): # Starts Bubble Sort Outer loop
                swapped = False # flag to detect any swaps
                for i in range(mx): # starts inner loop
                    if data[i][1] > data[i+1][1]: # compares scores
                        data[i], data[i+1] = data[i+1], data[i] # If sorting is needed, swaps the positions
                        swapped = True # if sorting needed, changes flag to true
                if not swapped: # checks if no swaps happened
                    break # stops loop

            for name in range(len(data)): # Rendering the names
                index += 1 # increments
                if index <= 6: # only renders top 5 scores
                    draw_text("{}".format(data[name]), 40, BLACK, WIDTH // 20, HEIGHT // 6 + (index*40)) # draws text onto screen

        pygame.display.update() # updates the screen

        for event in pygame.event.get(): # checks for event
            if event.type == pygame.QUIT:
                run = False # Closes the state
            if event.type == pygame.KEYDOWN: # checks if any key is pressed
                if event.key == pygame.K_RETURN:  # Checks if enter key is pressed
                    return START # returns to main menu
```

```
            if event.type == pygame.KEYDOWN: # checks if any key is pressed
                if event.key == pygame.K_RETURN:  # Checks if enter key is pressed
                    return START # returns to main menu


# creating a function for the help state
def help_screen():
    background = pygame.image.load('help1.png') # loading image
    background = pygame.transform.scale(background, (WIDTH, HEIGHT)) # transforming to preferre size
    run = True
    while run: # while run is set to true..
        window.blit(background, (0, 0)) # displays the imported background
        pygame.display.update() # updates the screen

        for event in pygame.event.get(): # checks for any events happening
            if event.type == pygame.QUIT:
                run = False # Closes the state
            if event.type == pygame.KEYDOWN: # checks if any key is pressed
                if event.key == pygame.K_RETURN:  # checks if any key is pressed
                    return START # returns to main menu

# main game function
def main():
    clock = pygame.time.Clock() # helps control the game's FPS
    state = START  # program starts in the start state``
    score = 0

    while True:
        clock.tick(FPS) #sets the the FPS at 60
    # The following code call the needed state
        if state == START:
            state = main_menu()
        elif state == PLAY:
            state, score = play_screen()
        elif state == GAME_OVER:
            state = game_over_screen(score)
        elif state == LEVEL_SCREEN:
            state = leaderboard_screen()
        elif state == LEADERBOARD:
            state = leaderboard_screen()
        elif state == HELP:
            state = help_screen()

        # closes application if user wants to quit
```

```
# main game function
def main():
    clock = pygame.time.Clock() # helps control the game's FPS
    state = START  # program starts in the start state``
    score = 0

    while True:
        clock.tick(FPS) #sets the the FPS at 60
    # The following code call the needed state
        if state == START:
            state = main_menu()
        elif state == PLAY:
            state, score = play_screen()
        elif state == GAME_OVER:
            state = game_over_screen(score)
        elif state == LEVEL_SCREEN:
            state = leaderboard_screen()
        elif state == LEADERBOARD:
            state = leaderboard_screen()
        elif state == HELP:
            state = help_screen()

        # closes application if user wants to quit
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()

# makes sure the games only runs if the game is called directly
if __name__ == "__main__":
    main()
```