**Contents list** **- Analysis**

Description –

Stakeholders –

Interview plans –

Interview conducted –

Analyse the interview –

General research into similar games –

Similar problems –

Justifying my approach –

Further meeting with stakeholders –

Users’ needs + abstraction diagram –

Stakeholder requirements –

Hardware and software –

Functions/ successes of program –

**Design**

Breaking the problem down – page 19

Structure of the solution – page 20

Justifying the usability – page 21‐22

Complexity – page 22

Algorithms and justification – page 23‐24

Code structures – page 25‐26

Tests to be performed on the iterative test – page 27

Post development data –

**Making and Test**

The layout of unity –

The player controller –

Sprite renderer and collider –

Idle animation –

Animation transition –

Movement code -

Animation and jump –

Jump problems –

Jump problem fix –

Jump code and animation –

Blend tree –

Double jump limit –

Maximum vertical velocity –

Improving the current code –

Fixing the improved code –

The enemy slime –

Slime’s jump attack –

Enemy and player health reduction –

Player destroy script (Crash fix) –

Better AI for slime –

Projectile attack for the player –

UI code –

UI fixed –

**Post testing:**

Post testing –

Usability test –

**Evaluation:**

Cross reference with success criteria –

Justifying unmet criteria –

Usability evaluation –

Maintenance and limitation –

**Bibliography**

**Analysis**

Description

My game will be a tactical game in which it can be played by any age from 4-16, it is a game aimed at a lower age range as it provides simple puzzles and small obstacles that need to be overcome in order to progress. To control the character chosen by the user there needs to be an input of some sort, for example a keyboard click or mouse input, in response to this there must be an output on screen to provide visual feedback, therefore this suggests there needs to be an input/ output mechanism to play this game. When I am writing the game I will need an IDE, which provides an environment where I can run and type my code. This sort of program will allow me to test and debug my code as well as input subroutines to input extra functions and variables into the game e.g. coin number or ”Number of lives”. The software this type of game needs will vary with different types of computational methods, for example displaying video feedback.

In addition to an IDE a game engine will be needed to be used to provide the extra functionality of this game compared to writing it in a regular IDE. When testing the game I will use the screen, keyboard and mouse used to type the code. In parallel with writing my game I will be reading up on the language I will be writing in, this will be python.

When designing this game there will be some obstacles that I will need to overcome such as glitches in the animations or the health etc. To overcome these problems I can reuse code that I have previously written or any working code that is on the internet. These will be copied into Notepad++ which is a good IDE I can use to write my code, it provides more functions than notepad by allowing a debugger. Also other programs similar to this have compilers and interpreters.

The user will need to move the on screen character to avoid obstacles, enemies and collect in game currency that can be used to purchase upgrades in the in game store. The game will be 2D and the character will be able move up and down, and increase and decrease speed.

To code these variables they will have to be assigned into a subroutine. When compiled it will pass these by reference so when they are changed it will not affect the main program. When the program is run, the computer will do the logical operations of the game, these are calculated in response to the users input, allowing for a fluid exiting game.

The Stakeholders

The ideal target market for my game would be an age range from 4-16, it is a simple no violent game, which can be played by young children. The game will be in 2D, which will reduce the time taken to code, because the character model will not need complex 3-dimensional animations, therefore small movements can be animated and assigned to any enemies or in game characters, however this will not change the experience of the game. Due to the lack of resources available in our school, I do not have a very powerful computer, therefore by coding with 2D models, it would not require a vast amount of resources to make and run, therefore this game will be available in primary schools/ on low budget PC’s.

The stakeholders will influence my game in many ways, the puzzles involved within the game should not be too demanding/ confusing. In order to achieve this level of difficulty, I will constantly review the game with my stakeholders by using a sample of children from 4-16and work on their feedback in order to improve the game.

The stakeholders I am making the game for will benefit throughout the process as they will get to test it at different stages which will allow them to give accurate feedback on the game. With the end product the stakeholders will be able to play the game again to see how it has progressed and developed into the game it is. By testing frequently it will allow me to go back through the development cycle and add, remove or change any features in the game.

The output of the program is very useful to them, without it they will not be able to play/ test the game, therefore the way the game is output is very important. I have thought of many ways it could be output and I need to make a decision on which I think is the best one.

One example is – using great pixel art to simulate good graphics, this would take a lot of time and investment into art/ design of the game. However having too high graphics would put a lot of pressure on the computers which are relatively low budget. The lack of a graphics card would put a lot of strain on the computer and some graphics may not be output, therefore the art should not be extreme.

Another example is having a quick response time, therefore a fluid game is simulated. In order to reduce input and output lag as much as possible to code much be kept simple and compact allowing the computers to compile the machine code at a much quicker rate. Also by reducing the amount of enemies and coins shown on each frame, it will reduce the time taken to load the entities.

Interview plan –

“Would you be interested in an upgrade feature for your character, or do you believe it will over complicate the game?”

I will ask this question to find out whether there is much point in spending time and resources in coding this extra function into the game, if there is a mainly positive response then I will add it in. The point of this function is to allow the player to enhance the user experience. Before is start coding this I need to decide on what upgrades will be available, I will not add too many extra functions as it will take too long to code, it will also cause more bugs which will reduce the fluidity in the game and the user experience. An easy way of enhancing the characters performance is by altering its speed and jump height, because these would be variables only the number assigned to them would need to be changed, therefore passing the variables by reference would have to be done so it changes the main program and changes the jump height or speed assigned to the character.

“Would you like to be able to kill the monsters in the game or just have to jump over them?”

The point of this question allows me to decide whether it would be worth my time adding in a health feature for the monsters and a hit point per attack on the character, this again could easily over complicate the program and would draw me away from the overall objective for the game.

“What kind of background would you find appropriate for the program?”

Due to the limited time I have to spend programming the game I will not have a lot of time dedicated to making pixel art, therefore I will need to re-use one background. Pixel art is very time consuming, with this and adding the backgrounds that would be randomly generated to the game would take too long to code. By having the input of potential users it will allow me to choose which background would suit the game best, it would also allow me to choose the one the majority would like.

“Do you want the enemies to be able to move or just shoot poison at your character?”

Again due to the limited time I have coding I will need to take into account what the stakeholder wants compare with what is actually possible in my time frame. In my opinion It will be more of a challenge and will enhance the users experience with the game if the enemy just shot poison at the character, this will provide more of a challenge, as the user will have to jump and avoid more obstacles to continue on with the game.

“Would you like the feature of upgrades for your character?”

I will ask this question because I feel it may over complicate the game and the user may just want to play the game as it is and not bother with any upgrades. However if the stakeholders decide they do want upgrades i will only offer a small amount they can be used, for example by altering speed or jump height would be easy as the variable assigned to them would be changed, modifying the characters performance.

“Do you want the game to have a currency system?”

This question will probably be yes, because it will add extra functionality to the game, also in order to add an upgrade system for the characters an in-game currency will be needed to purchase the upgrades. Also collecting in game currency will be an incentive for players to keep playing.

When I am doing the questions I will complete multiple interviews with different stakeholders, I will take a sample of stakeholders from different age and gender categories, and this will provide an accurate representative sample of my target market. Therefore my game will be best suited to the people who will play the game.

The interviewees are a random sample taken from different schools in my local area, their names are: James (age 14), Tia (age 16) and Jess (age 8)

Analyse the interview

“Would you be interested in an upgrade feature for your character, or do you believe it will over complicate the game?”

James: “I think the upgrade feature would be great to include in the game, it would really draw in the target market this is aimed at, as well as provide a great user experience.”

Tia: “I think an upgrade feature will not over complicate the game, most games have them, however most do not need them, nevertheless I believe it would really suit this game and complement its core aims.”

Jess: “It could over complicate the game if there are too many possible ways of modifying the characters performance, however by only having options for speed or ump height I think it will be a fun feature to have.”

“Would you like to be able to kill the monsters in the game or just have to jump over them?”

James: “I would like the ability to kill the monsters by jumping on top of them, I think this will be a fun attribute in the game, just jumping over them could become repetitive and boring. Having an option to kill them could also help you progress through the game.”

Tia: “Just jumping over them would be fun, however being able to kill them would mean they do not become obsolete obstacles.”

Jess: “If they are trying to kill you it would become exiting trying to kill them, it’s like you will be playing against someone, racing to who can beat who first, providing a sense of accomplishment when playing.”

“What kind of background would you find appropriate for the program?”

James: “Any background that suits the design/ graphics in the game, probably something like a cave or volcano. The background is very important as it makes the user feel like they are” actually there by setting the scene.”

Tia: “The background must match the theme, so I think a forest or cave like background would go well.”

Jess: “Monsters are quite scary which are found in scary places, so any scary background would be good. Something dark, and cold or really hot like the inside of a volcano.”

“Do you want the enemies to be able to move or just shoot poison at your character?”

James: “Shooting poison would be fine, it would create an objective to avoid, keeping you thinking throughout the game.”

Tia: “If the enemies moved it would take too long to code and could become annoying as they constantly are in the way or chasing you.”

Jess: “If the enemies moved it might be too hard for the computer to compile the code as they will be low budget PC’s. Therefore just shooting poison would be ideal. Also it would take less time and resources to animate the enemy if I just shot poison.

“Do you want the game to have a currency system?”

James: “Having a currency system would be good because it would complicate the upgrade system. When playing games I lie something to play for and collect, it makes me feel like I’m doing well so I know the more coins I have the better I am at the game.”

Jess: “A currency system would be great! I like the idea of collecting coins while trying to survive.”

Tia: “I would like a currency system but I only want to collect them by hitting them when they are on the floor.”

Results of quiz –

“Would you be interested in an upgrade feature for your character, or do you believe it will over complicate the game?”

The results of this question was a positive feedback for adding an upgrade feature with all of the interviewees on board with the idea.

“Would you like to be able to kill the monsters in the game or just have to jump over them?”

The response to this was as expected, most of the interviewees preferred to kill the enemies by jumping on top of them, a select few came to this conclusion suggesting killing them any other way would overcomplicate the game, meaning the code would not be completed by the target date, in addition the game wanted to be creative, by adding this jump kill feature would definitely make the game unique.

“What kind of background would you find appropriate for the program?”

All three people suggested a cave like background to provide a theme that compliments the game, adding a scary ambiance when playing. The aim of this interview was to find out what the stakeholder wanted, as they are the ones who will be playing, therefore to keep them happy I will design and add a cave like background.

“Do you want the enemies to be able to move or just shoot poison at your character?”

Shooting poison at the character was a preferred feature, it meant the user skills will be tested avoiding the bullets of poison flying at the character. It also meant I did not have to focus the majority of my efforts into melee combat and animating both characters in the game. This would give me time to add extra features such as the currency system or add to existing details in the game.

“Do you want the game to have a currency system?”

Again the game currency had very positive feedback, it meant the upgrade system had a way of working and user had an incentive to continue playing to keep upgrading their characters stats.

General research on video games –

**A resource bar:** A resource bar is commonly displayed as “mana” or “energy”, however I will be renaming mine to “coins”, this bar will be updated by 1 every time the user collects a coin. The program will recognise when the user has walked over a coin and it will change the variable assigned to coins to increase by 1. The resource bar should be located put of the way at the bottom or top corner of the screen.

**A health bar:**  A heath bar is similar to a resource bar however as well as updating if the user picks up a health pack, the bar will go down if it is hit by the enemies poison. This again will be assigned to a variable, this will be passed by reference so it doesn’t affect the min program and set the variable as that forever.

**Combat:** Most games usually have some sort of combat feature, however I will not be including one in my game. The aim of my game will not be affected by a lack of combat animations, the feature of fighting enemies will still be there, however there will not be complex animations I have to make. The character will still have a dynamic movement but it will only change between one or two moves. This will simulate a fight but will also reduce the time taken for me to code and animate the game.

**Time limit:**  There will be a time limit on my game. Some people prefer this feature because it provides a sense of urgency to complete the level and adds in a competitive aspect to the game, where they can compete against people worldwide or locally against their friends.

**Platforming:** My game will heavily involve platforming… Platforming is a feature where the character has to overcome physical barriers in the game, most of my features are based around the game super Mario, where the user had to jump over pipes and onto platforms to collect coins, I will be taking this idea but removing the pipes in order to provide a unique experience. Also in Super Mario they had enemies to fight and jump over, however I will be altering their behaviour as they are static but fire poison at you instead.

Mario research

Mario is a simple 2D game, I have taken many ideas from this game and have implemented them into my game. It is made by Nintendo and is very popular within the gaming community. There are many sequels of the game, with different modes and levels that the user can go through because of its popularity.

I have similar goals to Nintendo when they made Mario and I hope I can hit these targets when I make the game. Mario was very successful and took off very quickly, I hope I can replicate this with my game – this shows it was a strong game idea and that it appealed to the stakeholders needs/ wants. Nintendo and I both have similar stakeholders therefore similar goals to impress and keep these stakeholders satisfied, and if I create a similar game then I will be able to hit the needs of my stakeholders.

The original Mario games had green pipes that the character could jump into and it would take them to a different part of the level, however these are highly recognised and assigned with the Mario game series, so I will not add them into my game due to copyright purposes. In addition to this to travel through the game there was enemies that walked around and platforms you could jump on – the enemies in my game have been slightly changed in appearance and in the sense that they will not move – they will have a more adapted role in my game – they will shoot poison at the character.

What is similar about my game and the Mario game is that when the Mario game series was first released they did not have powerful devices to play on, therefore the graphics were very limited. The stakeholders will not have very powerful computers either, and I do not have a lot of time to focus on making amazing graphics this is why the background will be a continuous pixel art. However the hardware will not be limited, they will have a monitor, full-sized keyboards and a mouse, which is all that will be needed to control the character and there attacks.

On the Mario games, they were regularly played on a Nintendo console which came with a keypad, however this could limit movement or if a combo attack was added to the game, the user wouldn’t be able to take advantage of this. In addition to this using a keyboard it would allow anyone that doesn’t know how to use a gamepad to still be able to use the game.

Likewise in Mario, there was only one character you could play with – this was Mario (the game was named after the character) however as the game evolved and adapted it introduced the function of being able to play with other characters that had been shown in earlier versions of the game – that Mario meets on his journeys such as Luigi.

In later versions of the game they offered a 2 player feature – this allowed teammates to complete difficult levels together. In addition to this in very recent years Nintendo announced a multiplayer version of the game coming out – this allows users to play with their friends on the same levels but on different consoles. However because Mario isn’t a PC game there would be much point in having a multiplayer game, as my game is targeted at a PC users, these tend to be at a desk or office setting, which does not leave much room for two people to sit and play, furthermore I will not be offering a multiplayer feature.

Justifying the approach

To make this game I will need to be very organised, games are very hard to make and there are a lot of steps to develop a game – there is so much behind the scenes work to be done when making a game, for example, market research, advertisement, planning the stakeholders. As well as this, games are an amalgamation of lots of different scripts that interact with each other. It is very easy for something to break the code, therefore I need to keep everything organised so if anything goes wrong I know where and what went wrong so I can fix it. To do this I need to name variables and files appropriate names also I will leave comments in my code explaining each function.

When I start coding my game, it is important that I make assets so that the user will be able to see the effects of the code if there is not graphics to see them on.

I had not though about adding in a power bar, however I think it would be a really easy feature to add, it would be very similar to the resource bar however when it goes down it correlates with how fast the character moves, and to gain more energy you collect energy points. This means to continue on the level the user needs to keep collecting energy points.

A combination of this and adding a time limit could push users to rush thought the level to try and complete it, this could add a level of difficulty as sometimes the user may not have time to think about what they do forcing them to make a mistake which could lead them to die and start the level again – this is where the idea of a competition comes into play – friends may compete to see who can complete the level the quickest.

I will be platforming in my game which will allow users to run or hide from enemies, it will also allow them to collect more energy points as well as coins.

Limitations –

There will be many limitations when coding the game, some things make take more energy and coding than I originally though therefore the quality will have to be sacrificed somewhere else, this might be in the quality if the pixel art, the user won’t mind about the pixel art as long as the game works smoothly and properly. In the future I could add an update to the game with a higher quality pixel art, however the game must be released by a certain release date.

Another limitation is that I will not be able to make different looking levels, all levels will have to look the same as I do not have time to paint multiple varieties of backgrounds – this may help me develop the quality in this one background. Again I do not think the user will mind if the background is super high quality or not, as long as the games functions work.

Furthermore a third limitation is the enemies in the game will all have to look the same – this is a similar problem with the background… as I will be designing them, most of my time will be going to developing the code and I will only be able to fit in time to design one character and repeat them throughout the level.

The game will have some sound affects however these will probably be very low quality as most likely they will be basic stock sounds taken from the internet or sounds I have made myself, due to the fact I do not have access to high quality equipment or a recording studio they will probably have to be altered in an audio editing software which could affect the quality. Furthermore the stakeholders will only have very basic speakers if any at all, the will probably be the built in speakers that come stock with the PC, which means the quality of the sound will not matter that much.

Further meeting with stakeholders –

**Should I add an energy bar into the game?**

James: “I think adding an energy bar would be a really good way of adding competition, it would make the user think about what they do next, if they have enough energy to get past an obstacle or whether they need to grab more energy points.”

Jess: “I think it would make the game really realistic as the user would have to focus on keeping their energy levels high just like real life.”

Tia: “This seems like a really good idea, however I hope it won’t affect the time the game is completed by.”

**Should I add a timer into the game?**

James: “Adding a time might make the game too hard because it would make the user rush to the finish line while worrying about what their energy level is.”

Jess: “Having a timer would be a great idea! It would cause friends to compete with each other who can complete each level the quickest – which would encourage more people to play!”

Tia: “The idea of a timer on each level could be quite rewarding – by creating personal bests and world record… Maybe a leader board feature could be introduced and people could compete to beat people around the world?”

**Should puzzles be introduced into the game?**

James: “I think puzzles would just ruin the game because it won’t really go with the theme or idea of the game.”

Jess: “I think puzzles that are themed like the game and fit in really well, say if a challenge was completed you had to pass a puzzle before you could continue.”

Tia: “Adding in a puzzle would take too long to code alongside everything else, therefore I don’t think they should be added.”

**Should the game have ways of platforming in it?**

James: “Since the game is based around Mario and Mario has introduced platforming in it, I definitely think platforming should be introduced.”

Jess: “Platforming would make the game so much better, by being able to take combat onto other levels would make it really interesting”

Tia: “By having platforms would mean there is a chance of missing some resources which would mean users have to think ahead and develop their skills when playing.”

Analysis: The stakeholders were very keen for me to add an energy bar, therefore I’m going to go with what the majority decide and add one. However they did not like the idea of a puzzle, therefore I will not add one, due to the fact it would take too long to code and it would not suit the game. Moreover there was a majority of positive feedback for platforming the game and adding a timer – The timer would increase competition and the platforming would make the game more fun etc.

Abstraction of the problem

The game

Game mechanics

Combat

Currency

Time

Terrain generation?

Upgrades

Energy

Life

Platforming

Items

Game Aesthetics

Backdrop

Parralaxxing

Characters

Item

Foreground

Assets

Animations

Lighting

Platforming

Upgrades

GUI

Combat

Movements

Background

Lighting

Game sounds

Combat

Ambient

Menu

Collision detection

Describing the abstracted problem –

Hierarchy line: This describes the biggest to smallest tasks that I will need to undertake in this project. This diagram just breaks up the tasks of the game so I can see what I need to do to make the game – it presents it in a broken up simple format.

The game: Three things branch off the “The game” section, these are the main things I will be focusing on, each section for example “Game Aesthetics” has smaller branches such as “Assets” and “Lighting” that come off it which are the smaller tasks that needed to be completed in order to make up the whole game aesthetics.

Game Mechanics: Game Mechanics are a huge part in making up the game, this is why it has so many sub tasks that need to be tackled in order for the game mechanics to be smooth – without them the game would not be able to run and the user would not be able to input anything. They are added to the software to make the game fun and all functions work properly.

Game Aesthetics: The games aesthetics determines how the game looks, therefore the lighting and assets need to be considered how they are made for example what shape the enemies will be and how they will be made out of pixel art. In addition to this, there will be pre-set animations/ combat moves the enemies will have, therefore these need to designed and made. The animations and asset sub-Hierarchy both have sub-Hierarchies themselves which come together to make them.

Assets: Assets are classed as anything physical in the game for example, any currency that could be collected, an enemy or an object that is in your path. The sub-Hierarchies are there to keep me organised when I am making the assets.

Animations: The second sub hierarchy animations is made up of 4 further sub tasks, these are Combat, Movements, Background, and Lighting. These are very important for making the game look realistic and fluid when the user is playing, without combat the game would be very boring as there would be hardly any input the user can put into the game, this would also make the need for any enemies obsolete. In a games the window size is a fixed size of pixels, therefore there needs to be some code that moves the background when the character hits a certain point in the screen, this creates an effect that the screen is bigger than it is, although the pixels are still a fixed size the illusion created is the player is moving around the world.

Game sounds: I am not experienced in game sounds so I will not invest a lot of time here, however I will input small sounds for example when the player gets hit by an enemy, I may also add some sounds for when the character lands onto a platform.

Stakeholder Requirements –

* Platformer elements
* Combat to be involved
* An upgrade system
* A currency system
* Animations
* Sounds
* GUI

Platformer elements: The platformer elements I will need to include will be basic things such as jumping and platforms to jump onto, in addition to this the background will need to move to the side in order to progress in the level e.g. to collect more coins and increase your score. With the platforms I will make it so the character can travels through the bottom of the platform but lands on the top of the platform.

Combat: I will have to draw multiple action poses for my character, by putting multiple drawings together it would animate it making it look like it is fighting, this will be triggered by a key press. While the computer is running the change in movement to simulate fighting it will be checking to see if the characters hit box collides with the enemies hit box, if it does then health points will be deducted from either character.

Upgrade system: When playing the game the user will collect coins which can be used as in game currency, these could be spent on upgrades for the character, for example a speed boost or jump boost. This will be easy to implement because it will only be changing a variable.

Currency system: Coins will be randomly located around the map to be collected by the user, they will be able to be spent in the shop which will be a window accessed on the main screen.

Animations: The animations take long to make therefore they will only be basic animations. The animations will be reused due to the strict time limits I have to stick to.

Sounds: Like I stated before I will not spend much time on the sounds in the game because I am not experienced in this area, however all games must have some sounds input into them to make the experience more engaging for the user.

GUI: The graphical user interface will be simple yet functional, it will mean the user can navigate it easily without any bugs and with all buttons displayed functional.

Hardware and software requirements –

Hardware: The stakeholders that will play my game will not have access to great computers therefore the hardware requirements will not be very hard to meet, this is because the stakeholders are children aged 4-16, therefore there budget is not huge, especially as they are in full time education.

The basic requirements will be a mouse, keyboard and a monitor that is capable of outputting sound, the requirements for the PC are listed below:

* CPU: AMD A8‐7410 APU
* GPU: AMD Radeon R5 graphics
* RAM: 8Gb @ 1600MHz
* HDD space: < 1Gb

Software: I am making the game using windows 10, however the software that are compatible with this game is:

Windows 7 +.

I am making the game using Pygame, which is an add on to python.

The language I am writing this in is python.

Success criteria –

**The combat should be fun:** In my opinion the most important thing a game should be is to be fun, it can become stressful and not a very nice playing experience when the game is too challenging, if this is the case the game might not be successful. To do this I will make sure the enemies the user are fighting are not too challenging but still make the user apply effort to kill them, this adds a rewarding feeling to the game, by making the user feel that they have accomplished something and are progressing in the game.

I would want to test this to make sure the game is fun and is not too challenging, however this will be hard as enjoyment is very subjective, with the sample of people I will use, some may find the game more enjoyable when it is more fun, however some people may find it fine the way it is and others may want it to be less challenging, and the purpose of that test is to find out what the majority of the stakeholders would like, in order to keep them happy.

**The main character should have at least 2 combat moves:** By having more than one combat move, they can be selected at random which would make the combat a lot more interesting and realistic. If there was only one move then it might become boring which is not the aim of my game. I could use annotated screenshots to prove the animations work in practice.

**There should be an enemy that the player can fight against:** By having randomly generated enemies it doesn’t just provide a sense of progression and achievement it allows the user to have fun in the process. The AI that I will develop should be good enough to take the characters health down to 0 but not very quickly, this should provide enough of a challenge for the user to have fun with and not get bored of because it is too easy. To test this out I will play the game a few times myself, trying to beat my high score each time and I will keep track of how many times I die, this way I know if it is too hard or too easy.

**The UI should be clear and display correct information:** It is very important that UI is clear and accurate so the user knows how much health they have left and how many coins they have collected. If the user knows how much health they have left then they can decide to avoid any enemies or go into combat.

**Upgrade system with aesthetic impact:** When the user goes to purchase some upgrades I want the menu to look good as well as be functional, I will benchmark every line of code, to make sure when the user purchases an upgrade that it deducts the correct amount of coins and that the upgrade sticks.

**The movement should work effectively:** I am not going to add a double jump system as users might be able to exploit this when in the game, therefore I will make sure the user can only jump when their base coordinates are equal to a platform top. I also want to make sure all movements are smooth and there is no lag, this will ensure a realistic feel to the game.

When in the air players will be able to use the arrow keys to move left and right to position their landing properly. As I will not be using a double jump system I can prove the jump feature works by using screen shots of my character moving from the ground up to a platform above it. This will only need 3 screenshots, one on the ground, one in the air, and one on the platform.

**Design**

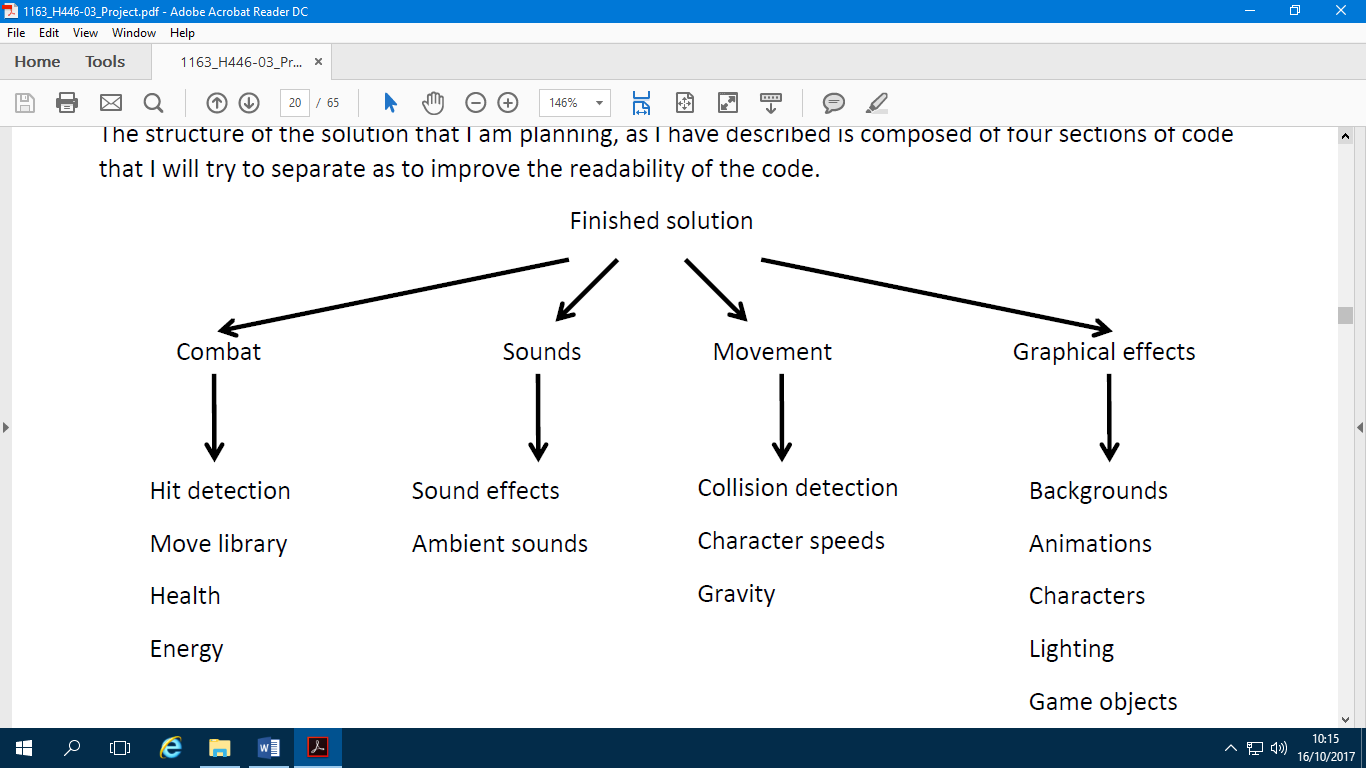
Breaking down the problem

I will need to keep the relevant code of topic all in one page or tab in python. I need to do this so I know where to find the relevant code if I run into any mistakes. There are four major topics that I will have separated out:

* The combat in the game will have many different subtasks such as: Damage, life, collision, detection and upgrades. I expect this part of the game to be the longest/ most complex part of coding as it can be quite hard to link all of these features together without some sort of bug at first. I will focus on collision detection at first so the character cannot just walk through the enemies, this will make the game seem unrealistic, also if collisions are not defined then the rest of the code for combat will not work, as the character would just move through the enemies. Another major part of the code will be coding in the different animations, however I feel that this would not bad as bad as coding in collisions.
* I will have a section of code with movement in it, with this will again involve lots of subroutines such as coding in some sort of gravity and velocity for movement forwards, backwards, and upwards, this would have to be implemented not just for the movement but to calculate when a collision would happen.
* In my code I will have separate files for things such as the sprite sheet, settings and the main section, they will all be linked together through Pygame. I cannot put all my graphical effects for the game in the main section as it will make it hard to read and fix any bugs, so I will link it with my sprite sheet, this makes sense as this is one of the main functions of the sprite sheet – to store the graphical information of the user and enemies, I could also add in my background – this should only be a couple of lines of code.
* I am not going to focus a lot of my time into adding sound or music to my game, I will add in a few sound effects to when the an attack is triggered, or when damage is done, but these will be very basic sounds, and because I do not have the resources to make my own sounds I will have to find some online that I can use in my code.

The movement, graphical effects and combat are all big tasks, because of their size and my limited time scale, I may have to split them into smaller subsections.

* Movement can be split into: collision detection, character velocity on the X and Y axis, and gravity.
* Combat can be split into: hit/ collision detection and health.
* Graphical effects can be split into: Backgrounds, animations, game objects e.g. coins or platforms, and characters

Structure of the solution

I feel that the code for my sections needs to be as efficient as possible, to achieve this in my main section is by using a lot of IF statements – this is more efficient than the usual way of coding 2D games where the processor checks the game logic on the constantly refreshing part of the code. This way I will only add the most important functions on the constantly refreshing main method. The conditional functions like hit detection will be assigned to IF statements so they are not checked on every frame, in turn this will reduce game lag.

The refresh rate of my main game code will be at 60 Hertz, I decided on this as it will provide realistic/ fluid gameplay, although my stakeholders will not have very powerful computers the game will not require a huge amount of graphical processing power or RAM as it is only a 2D game. This refreshing will check for key presses, each of which will be linked to a function e.g. Jump, hit or move forwards etc. Once the key is pressed it will call the function that is assigned to it which will then run the graphical subsection.

The process that will take the most time to make and the most processing power to run will be collision/ hit detecting, this is because the characters are always moving so there will have to be an accurate hit box around each character. In addition to this the refresh rate will have to be high enough that when a collision does happen it updates in real time. All potential collisions will be assigned a coordinate on screen and if the users coordinates crosses over with another block or character then it will be classed as a collision.

The enemies the player encounters will have their own coded attribute and fighting animations. The enemies will be very similar as I do not have much time to make lots of different animations, however I can make them look differently with different pixel art.

Features to be included

**Character movement:** I want the character movement for the main character to be consistent and responsive. The movement will be refreshed very frequently in the screen loop to make sure that it is kept responsive. I am planning to include a jump in the game, and I will allow the player to change the direction and velocity mid-way through the jump. It is very important to make the movement system of the game to be fun to play as it forms large part of the gameplay in the game, especially if I have enemies to avoid, they may have to change their velocity mid-air to avoid getting killed.

**Collision detection:** Collision detection is a huge part in my game because I will have to code in certain functions for the jump feature i.e. only jump once the player has landed on the platform. Therefore the character will have to have coordinate assigned to them and if their coordinates overlap with any other objects on screen then they will be known as colliding. This is why the coordinates have to be very close to the outline of the player to make it look as realistic as possible so two objects are not colliding when they aren’t visually touching. In addition to this the refresh rate needs to be high so it is constantly updating and checking for collisions.

**Combat:** I will spend the majority of my time on combat, there are many aspects that go into making a fluid combat feature in a game, for example the code needs to be there e.g. to activate the fighting moves, The graphics needs to be there to make a visual show for the user, this shows this action has actually happened and makes it look realistic. Things like keeping the animations short so that damage is inflicted instantly should be kept in mid because the user doesn’t want to wait after a button press for damage to be inflicted. The combat should also be kept responsive – it will not be fun running at 15 frames per second due to input lag. The AI also need to be coded for so that they are good enough to make a fun and challenging combat but not so hard that the user cannot defeat them.

**Hit detection:** In order for damage to be taken away from the player or enemies health bar, a function called hit detection needs to be added in, This is an algorithm that will decide whether the players coordinates overlap with the enemies or any other objects in the game. It needs to decide if it hits any objects, because if it does the computer needs to decide on an output. For example if the user collides with a platform then the computer needs to move the player onto the platform and out put the visual effects for this, However if the user collides with a coin then the algorithm needs to be run to remove the coin so it cannot be collected again and the amount of coins the user has needs to be increased by one.

Usability features to be included:

**The UI:** The UI is a usability feature, it will be showing the player the values that he needs to know, such as the players money bar and health bar. These bars will just be solid colours in a set rectangle to show the amount of health left compared to how much possible health you could have, this will indicate to the user when they are almost dead.

**The UI being described**: Before the game starts the UI may need to be explained to the users so that they know what each bar means and how they can use it properly. This could be added into the start screen so the user is prepared before they start.

**Mute:** A mute button could be added into my game, this will be to stop any noises from the player attacking the enemies or from the player falling off a platform from high up, this could be located in one of the top corners of my screen, however I will have to make sure it does not overlap with any of the UI’s.

I will not have any music playing in the background of my game therefore the mute button may not be needed.

Complexity:

This game will hopefully use a lot of mechanics that work together smoothly to add a lot of complexity and usability for the player.

The most complex part of my code will be the AI (Artificial intelligence which will follow the player around trying to target them by hitting them if they are to close) this is because the AI will be very hard to program and manipulate to get it to do what you want. In addition to this the AI needs to be coded so it is not too hard for the user to battle. The enemy will need to know if there are any objects in the way, when the end of a platform is and how to jump just in case it falls off, this can be very complex and can cause many bugs in the code.

Another complexity contributor will be the entire attack system for the player

Algorithms to be used

The algorithms I will be putting in my game are:

#HORIZONTAL MOVEMENT

def update(self):

self.acc = vec(0, PLAYER\_GRAV)

keys = pg.key.get\_pressed()

if keys[pg.K\_LEFT]:

self.acc.x = -PLAYER\_ACC

if keys[pg.K\_RIGHT]:

self.acc.x = PLAYER\_ACC

#VERTICAL MOVEMENT

def jump(self):

#jump only if standing on a platform

self.rect.x += 1

hits = pg.sprite.spritecollide(self, self.game.platforms, False)

self.rect.x =+ 1

self.vel.y = -PLAYER\_JUMP

#APPLY FRICTION

self.acc.x += self.vel.x \* PLAYER\_FRICTION

#EQUATIONS OF MOTION

self.vel += self.acc

self.pos += self.vel + 0.5 \* self.acc

#PLAYER PROPERTIES

PLAYER\_ACC = 0.5

PLAYER\_FRICTION = -0.12

PLAYER\_GRAV = 0.8

PLAYER\_JUMP = 20

To make the complete solution I have described that the game needs to have a movement system, therefore the algorithms for the horizontal movement and jump algorithms are necessary. Along with this movement code, I will be flipping the sprite of the player which comes under graphical effects that I will need to include In‐game. I will be animating these actions in‐game, to do this a loop in my code will be used to change the animations for the player, however the enemies will be constantly moving across screen and if the player collides with one the player will die. The game also requires a combat system for the player, I have added a collision detection system, which will pick up if the player or enemy has been hit, taking off the appropriate amount of damage. The enemy AI code is there so that the player will need to fight against the enemy. If the enemy is not ‘smart’ it may end up moving off the edge of the map, or it may not end up following the player correctly.

Identifying key data structure and variables

PLAYER\_ACC = 0.5

This is a very important variable in the code, it is assigned to a key press, for example if the player presses the left or right key then they will move in that direction with a velocity of 0.5.

PLAYER\_FRICTION = -0.12

This variable works with the velocity, when the key is pressed to move instead of instantly going to that speed a negative velocity is making the speed seem gradual, making the game more fluid and realistic, this is also the same when slowing down, this player slowly comes to a stop instead of stopping instantly which is unrealistic.

FPS = 60

This variable stands for “Frames per Second” it is defined at 60, this means there will be 60 frames appearing on screen every second. I chose 60 instead of a higher or lower number; the computers that will be use won’t be a high spec and therefore the monitors they are using won’t be capable of outputting any higher, there will be no need to put extra stress on the computer be running at a higher FPS. I also did some research and found that most games run at this amount of frames per second, it allows for fluid realistic gameplay without stressing the computer too much.

class player(pg.sprite.Sprite):

This is not a variable but is a key data structure, it holds lots of subroutines that link to the variables in the settings. It defines the player’s position and sets the velocity of the player to 0 when you first load the game. Later an “if” statement is added to assign the velocity to the player in whatever direction the user wants to go, defined by the arrow keys. This is added into the update section for the game, the game will constantly update to check for things like this in case the user wants to change their mind and go a different direction.

class platform(pg.sprite.Sprite):

This class randomly places the platforms on the screen and it imports the properties from the update section in the main, where it finds the player coordinates and if it crosses with a coordinate of the platform then it will detect a collision.

if event.type == pg.KEYDOWN:

This key bit of code determines whether the event “Keydown” has happened, the code that comes after this will run a loop that will increase or decrease player velocity in a certain direction and will run the animation for movement. It is found in the update section of the code.

self.screen.fill(BGCOLOR)

This fills the background of the screen with the variable “BGCOLOR” this is determined in my settings at I have assigned it a value of light blue (24, 144, 249)

self.mob\_timer > 5000 + random.choice([-1000, -500, 0, 500, 1000]):

This line of code determines how quickly and how often enemies appear on screen, I used a bit of a random number generator, so the algorithm can randomly choose a time it wants to spawn a mob, ranging from 0-10 seconds. This means the user cannot predict where and when the mob is going to come from, making the game slightly more challenging.

self.draw\_text("NEW HIGH SCORE!", 22, WHITE, WIDTH / 2, HEIGHT / 2 + 40)

This prints some text in a white font size 40, with coordinates of where to place it on the screen, it uses the height and width of the screen and divides them by two to make some coordinates to place it, this ensures it is in the centre. This only appears after the user has died and their current score is greater than the best score that is saved, so it uses some conditions to ensure it appears at the most appropriate time.

pg.mixer.music.load(path.join(self.snd\_dir, 'Happy Tune.wav'))

self.run()

In order to play music while my game runs I needed to add a bit of code into the game and reference what song I wanted to play. This is located into the part of my code where a new game is initialised, this is so it plays as soon as the game starts.

self.clock = pg.time.Clock()

What this code gives is the game the opportunity so it knows what FPS to run at as well as knowing when to quit the game.

self.wait\_for\_key()

I added a loop to the movement section for the character, it is constantly waiting for conditions to be met in order to output the animations and the velocity on screen. If this condition is met it will run the appropriate code for what the user wants.

if sprite.rect.bottom < 0:

sprite.kill()

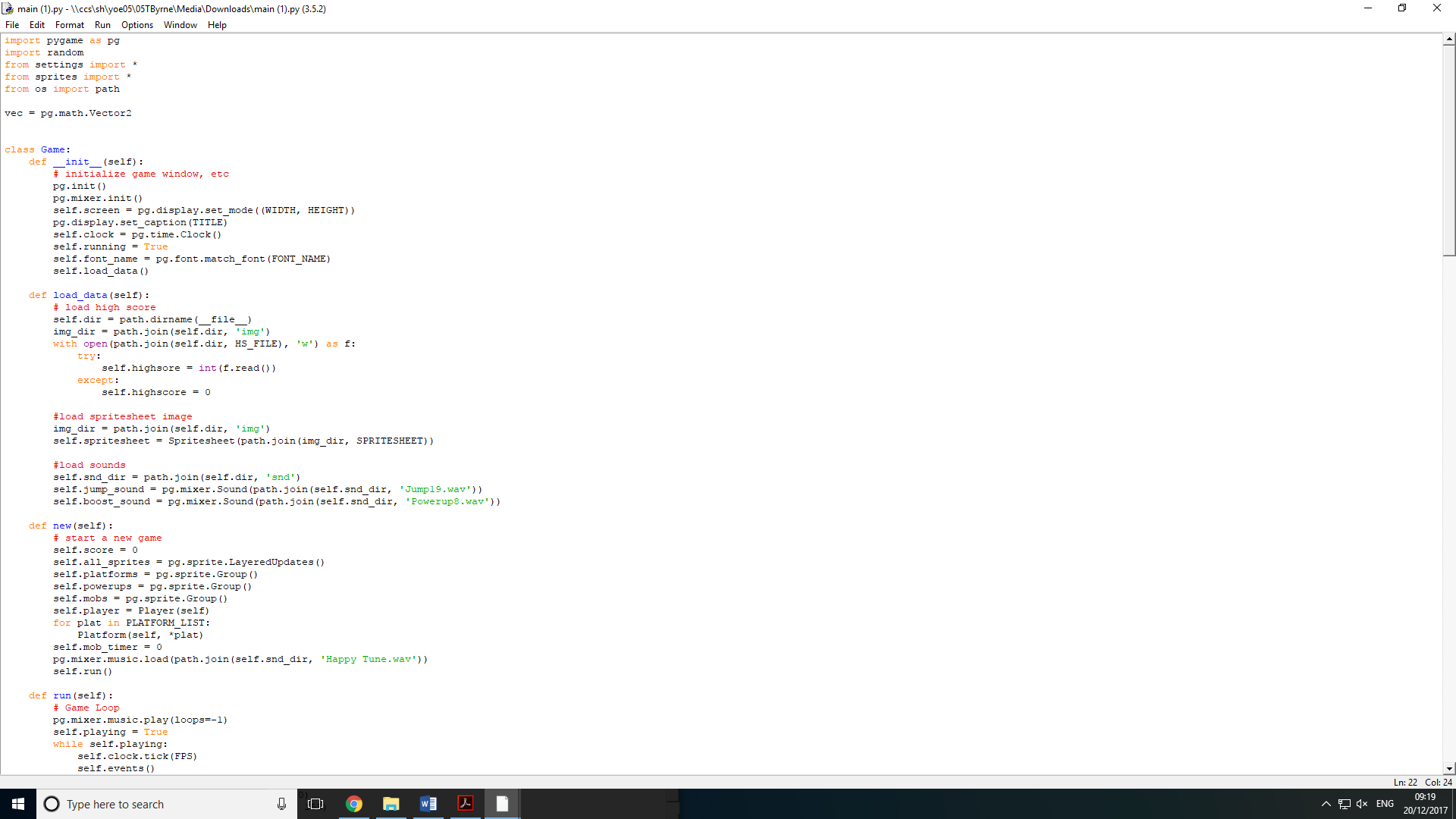
When I was designing the game I didn’t want the player to be invincible or play for ever, this can become boring very quickly, therefore by adding in the fear of falling to the bottom means the game ends it gives an incentive to the user to be cautious with each move and think about where they are going to jump to next. By achieving a high score this way it becomes more of an achievement than just a number, this creates competitiveness between users with who can get the best score. In order to achieve this, I wrote, if the bottom coordinates of the character is less than the pixels of the screen then they end game screen should be shown and the players health should be 0.

|  |  |
| --- | --- |
| **Tests to be done in the iterative development of the game** | **Justification of the tests** |
| Collision detection of the objects in the game | This could be tested by playing the game to check for any errors in my code e.g. if the player falls through the platforms when their feet went above the top coordinate. I will also be testing to see if the hit detection on the enemies is correct, it needs to be very precise to make the game seem more realistic and fluid. |
| Testing gravity in the game | This can be tested by seeing if the player falls down after jumping, I can also change the downwards velocity on the player in the settings to find the most accurate representation of real life. |
| Animation testing | This is easily tested by going into the game and using all possible buttons that should be associated with the players actions, if one doesn’t work I can go into the code and see why. |
| Testing the players horizontal movement | The player needs to move left and right at a speed that is controllable and is logical for the game, by going in and moving the character about I can increase or decrease velocity to an optimum speed. |
| Floor collision variable | So the player cannot cheat I need to use a Boolean value for jumping. This prevents the player from jumping in mid-air, if the player is not touching the top of the platform, then they should not be able to jump. Or if the players Y velocity is > 0 then again they should not be able to jump. This should be a simple algorithm checking whether the player has collided with a platform. |
| Testing the jump | The button I have associated with jumping is the space bar. If I go into the game and press the space bar and the character moves up and the jumping animations are shown then I know this is working. |
| The enemy needs to move across the screen | When the game is loaded mobs should move across the screen above the user so they are in the way and users have to navigate around them. Just by running the game I can see if this is working, if mobs appear then no more work needs to be done. |
| The enemy needs to attack the player | If the player bumps into the mob then the game should end and the game over screen should be shown, I can test this by jumping into a mob and seeing if this happens. |
| Players attack needs to hit the enemy | Precision point programming needs to be used to ensure the visual aspect of the collision is accurate so they are not colliding far away from each other. A mask will ensure when the characters collide their graphics are actually touching and there are no glitches. This can be tested by hitting the mob multiple times. |
| The UI should display correct information | I want to make sure the correct amount of health is taken away from the health bar for example colliding into a mob or falling from a height onto a platform, I will test this by jumping onto platforms for a varied amount of heights to see how much is deducted from each height, e.g. its’ not too much or too little. |

|  |  |
| --- | --- |
| **Post development tests to be done** | **Process of testing/ criteria to be met** |
| Horizontal  movement works | I will test this by pressing the right and left arrow keys on the keyboard.   * The player should move in the correct directions depending on the key press. * There should not be a lot of input lag. (the movement should feel instant) |
| Jump works | This test will be performed by pressing the space bar.   * The player should be able to do a double jump. * The player should move vertically with both of the jumps. * Not too much input lag in the jumps. * Spacebar presses should not be ignored. |
| Enemy paths to  Player | The player should move to a platform and wait to see if the enemy reaches the player.   * The enemy will successfully reach the player * It should take a straight route to the player |
| Enemy attacks  Player | The player should stand near the enemy and wait for it to attack you.   * If the player has a contact attack, the health should be reduced if they are touching. |
| Player can attack Enemy | The player will move close to the enemy and should press the ‘attack’ key.   * The health of the enemy should get reduced if in range. * The enemy health should not be reduced if the attack is not in range. |
| UI displays  Information | The UI bars should be filled in corresponding to the values of the variables. |
| Animation works | This will be seen through observation.   * Player movement animated * Player idle animated * Player jump animated * Player attack animated * Enemy jump animated if applicable * Enemy attack animated |

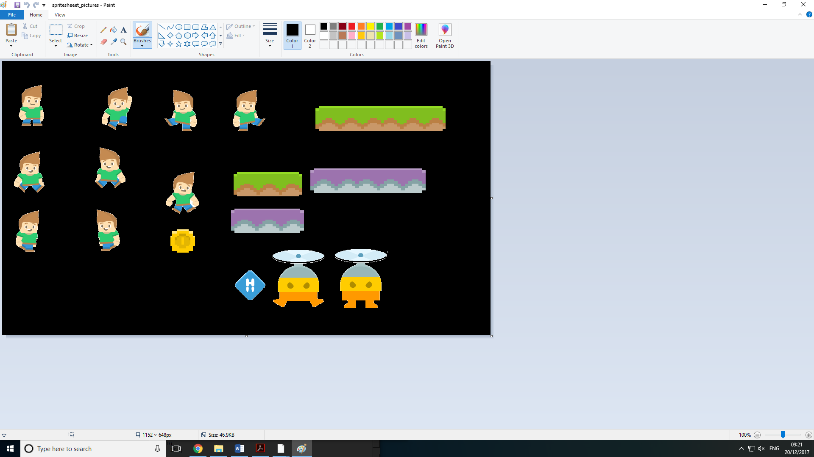
Making and testing

The layout of Pygame:

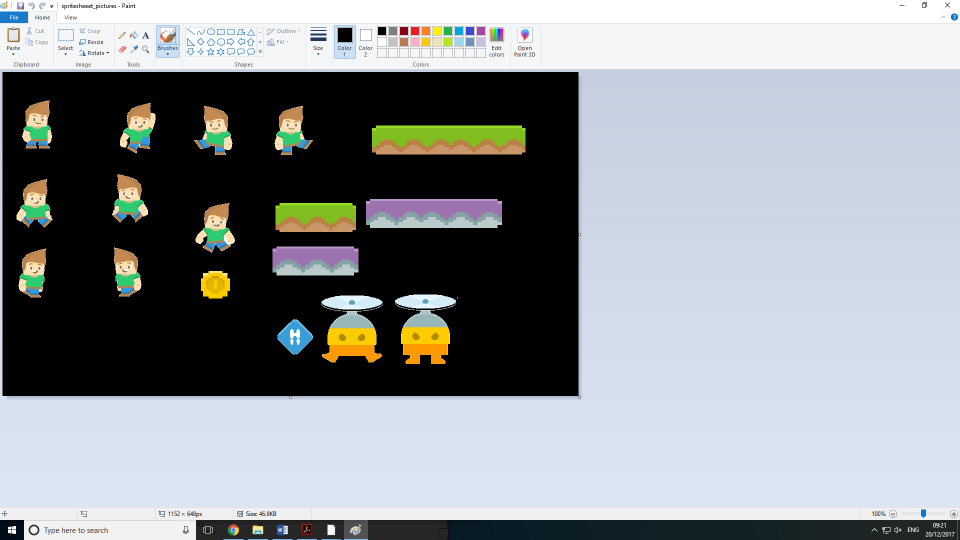


This is the environment where I will write and run my code, it is very basic and does not have many features to help in the writing of my code e.g. sentence guessers. Only 2D games and programs e.g. tuples or 2D arrays can be made using the python IDLE. I will only be making a 2D game so this is suitable for my game. In my game, when writing the code in order to make the game look realistic, I am going to be using layers, this ensures any features I add they will not clash, i.e. the clouds I am adding are in front of the blue sky but are behind the player, mobs and platforms. I have set the players and mobs on the same layer so when they collide they look like they hit each other and don’t overlap one another. The “camera” that will be in the game will be coded to follow the player as they continue up, I can add a statement that continuously checks if the player has reached a certain point on the screen and if they have more platforms will be spawned so the player does not go off the screen and can continue moving up.

To make my character, I will use pixel art and then move the images over to a spritesheet, then I will box off the spritesheet with lines at set coordinates so it is easy to add them into my game. The spritesheet will be linked in the code and coordinates of the position and size of each character/ stance will be given so when it is run it can find the image easily. This will be found in a folder full of all the things I need to make my game such as sound files, images, animations etc. I did not have time to make the images so I used a website online which has different artists that allow you to use their work for free. To make the sound files I used an application that has pre-set sounds for each different command or feature that might be in a game, it comes with a user interface that allows you to adjust the tone, frequency or duration of the sounds which is what I did to make a unique sound.



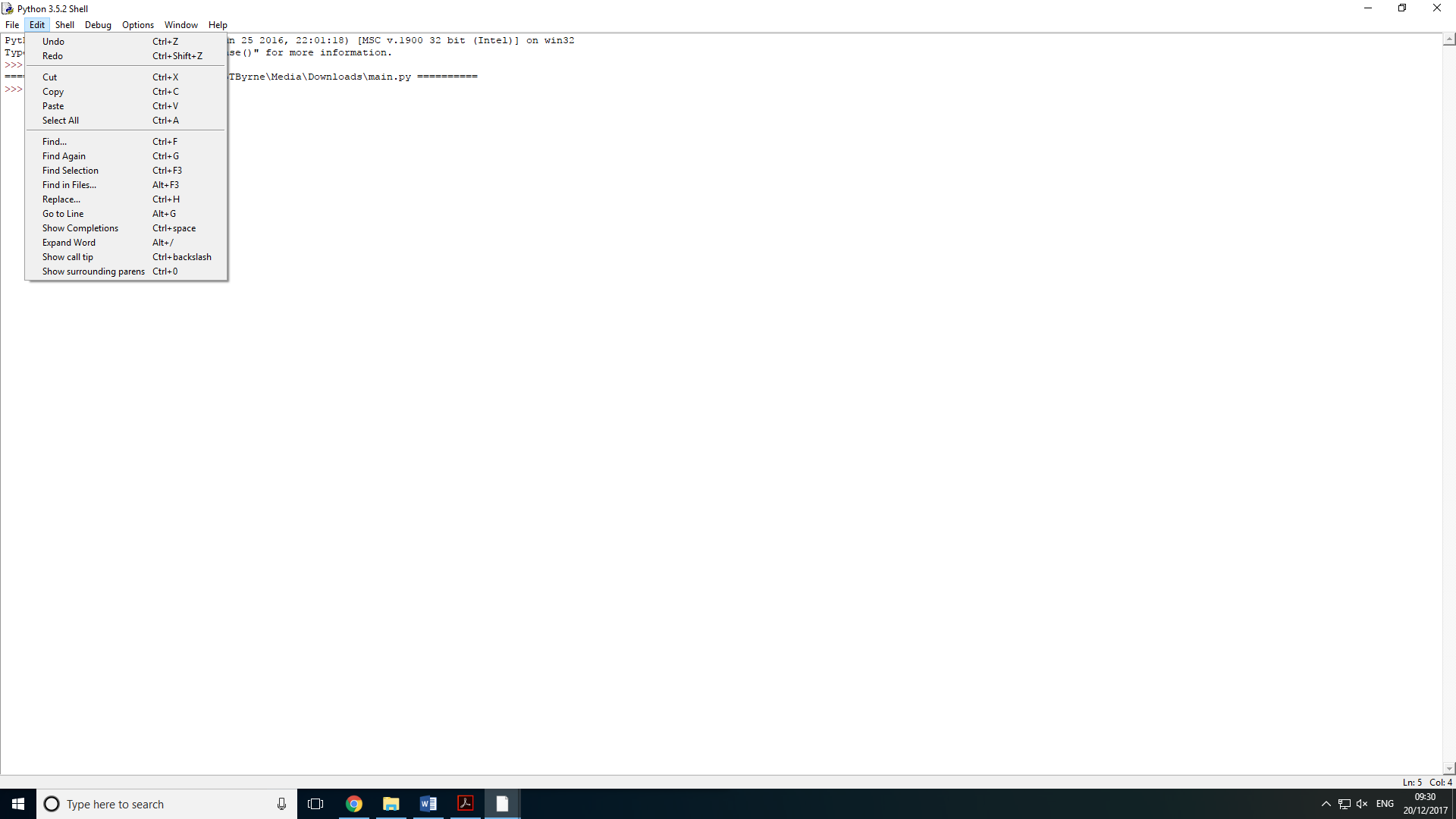
Here is the console log, it is where the code is ran and debugged so any errors in the code will be printed here. It tells me on what line the problem is located and gives me an indication of what it is. It is extremely useful because it tells me why the game is crashing or what is not working as intended.



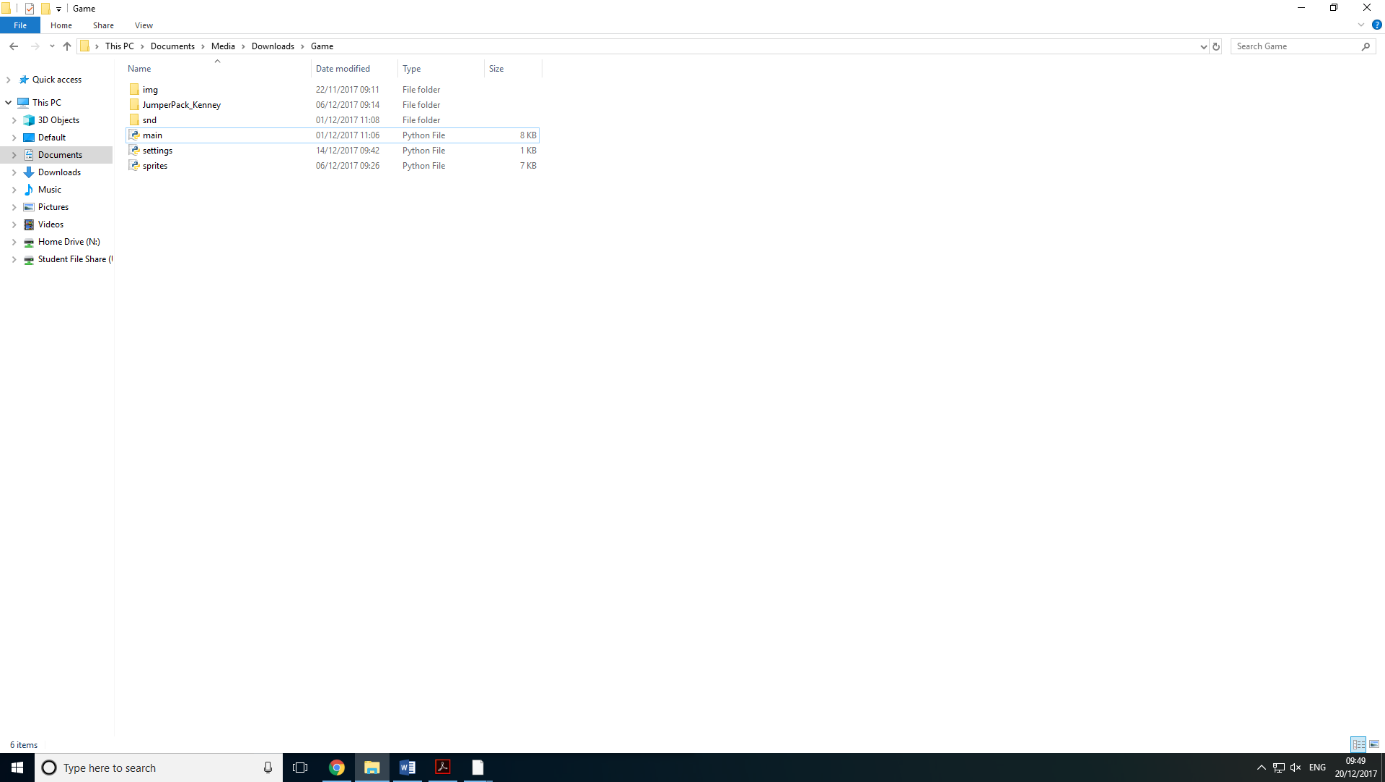
Python layout explained further

In the Python IDLE page, it has a drop down menu with many functions on it, an example would be the debugger, once you have input your code here and you hit the debugger function and a compiler will go through your code all at once and tell you if there are any problems with the code and where to find them.

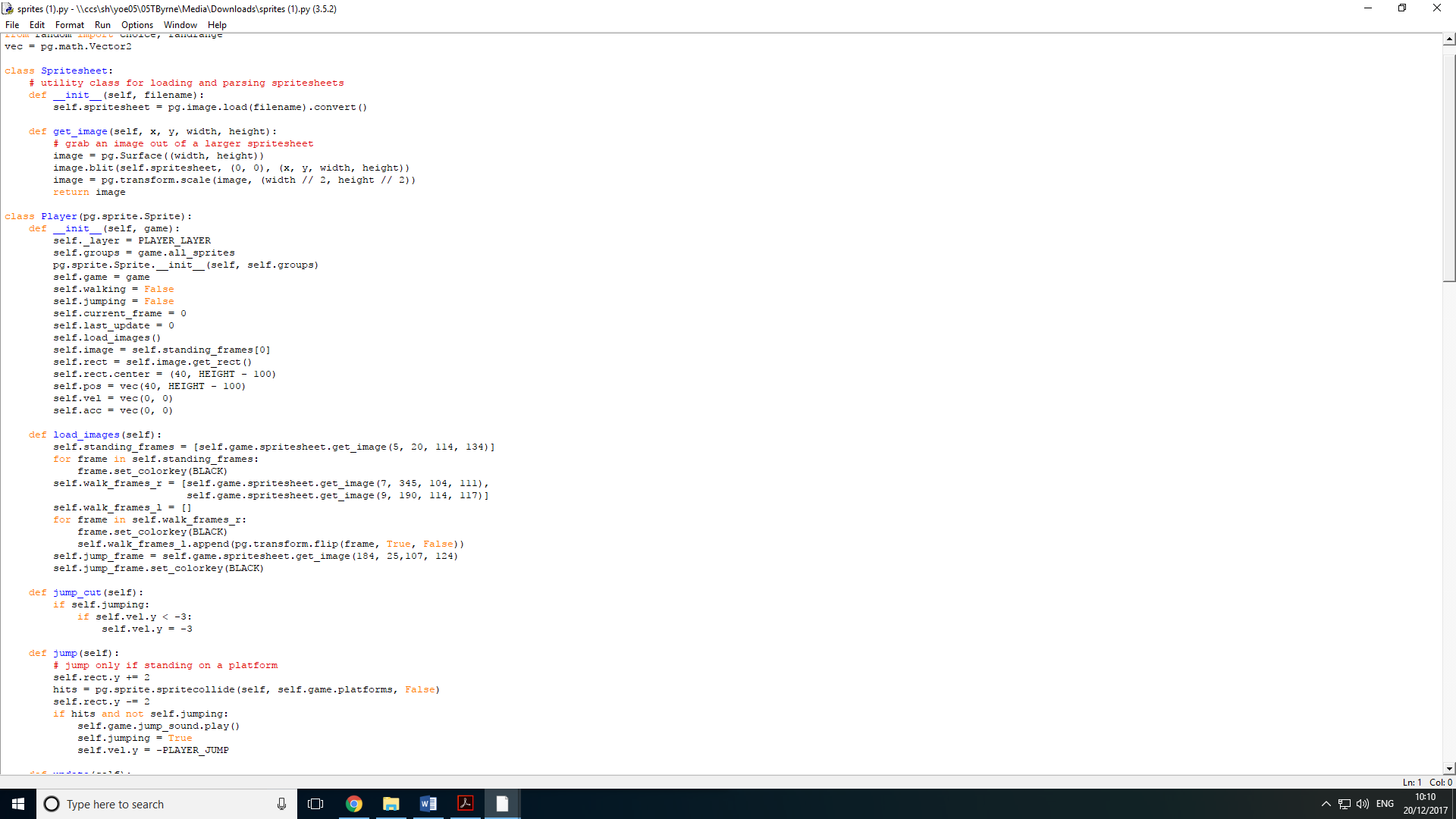
Another function is “Find”, this allows you to search for any word or phrase in your code. This will come in particularly useful if the console log says there is a problem e.g. a syntax error but you cannot find where it is.



Python doesn’t have any pre set components that can be added in as you are coding, this makes it much harder to make the game as I will have to make everything from scratch and develop everything myself. If I get stuck then I could follow an online tutorial or go into a forum and see if anyone is having the same problem as me. Another way to fix an error in my code could be through trial and error – changing small bits of the code that the console has said to change to try and get the desired outcome.

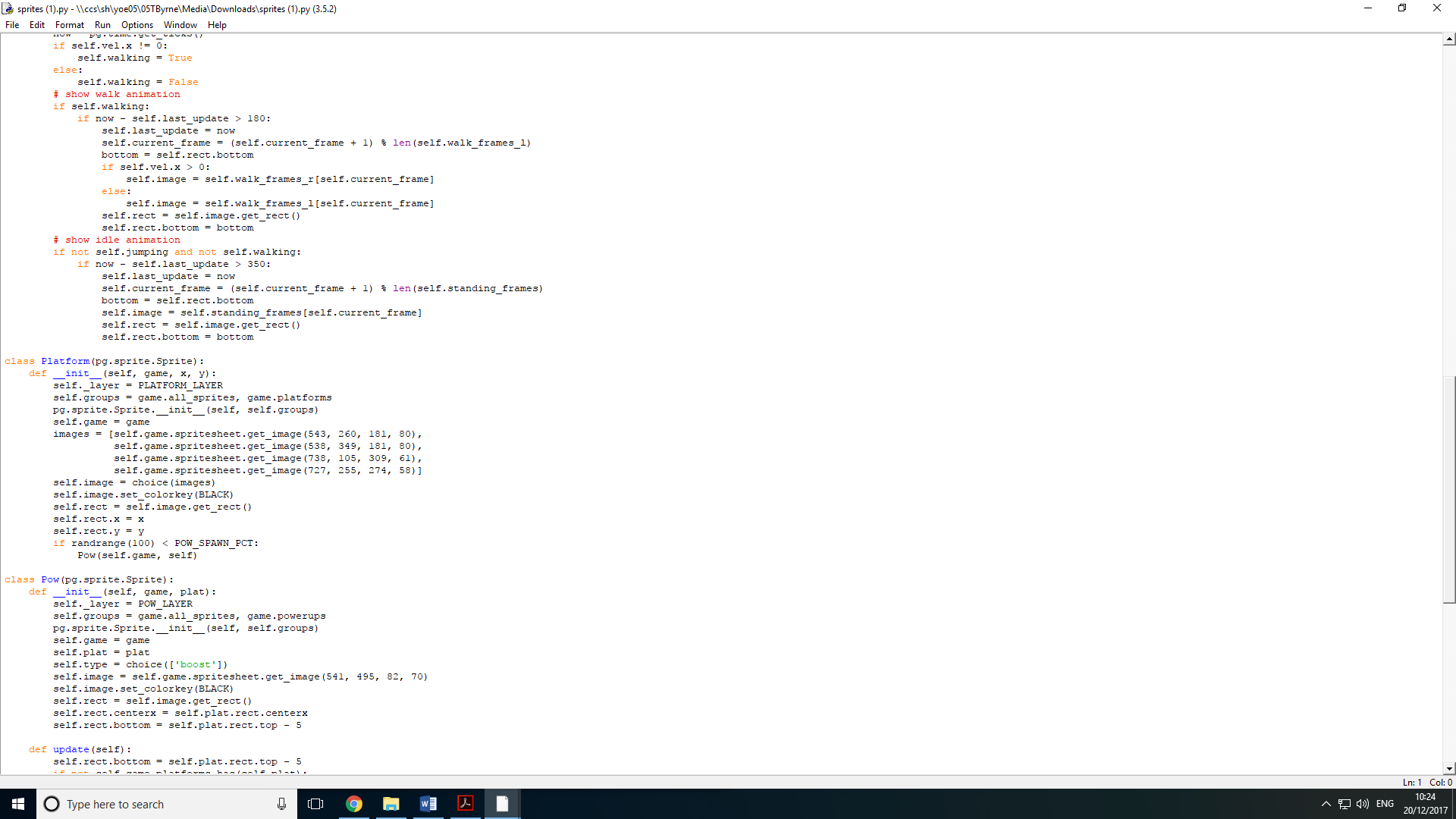
The “Game” folder is full of my code, spritesheets and sound files. These are all files that I have either created or found online and are in the correct format ready to be added into my code.

The player controller

To begin making the character in game I need to make a player class and define it, I also need to make an empty game object. I just need to add a few basic game constructs with a few simple lines of code. The game object will be referenced when making the player object, this is so the game knows to add the player in when the code is run. Also in this class I will list all the different features the character will have, such as: Walking or jumping. These will both be set to False as when the game loads up the characters state should be standing as they will not be walking or jumping. This will mean the correct image will be shown when the game is loaded – if self.walking = True then the walking image will be shown. This class is also where I will reference the standing image for the character and what size the pixel hit box should be.

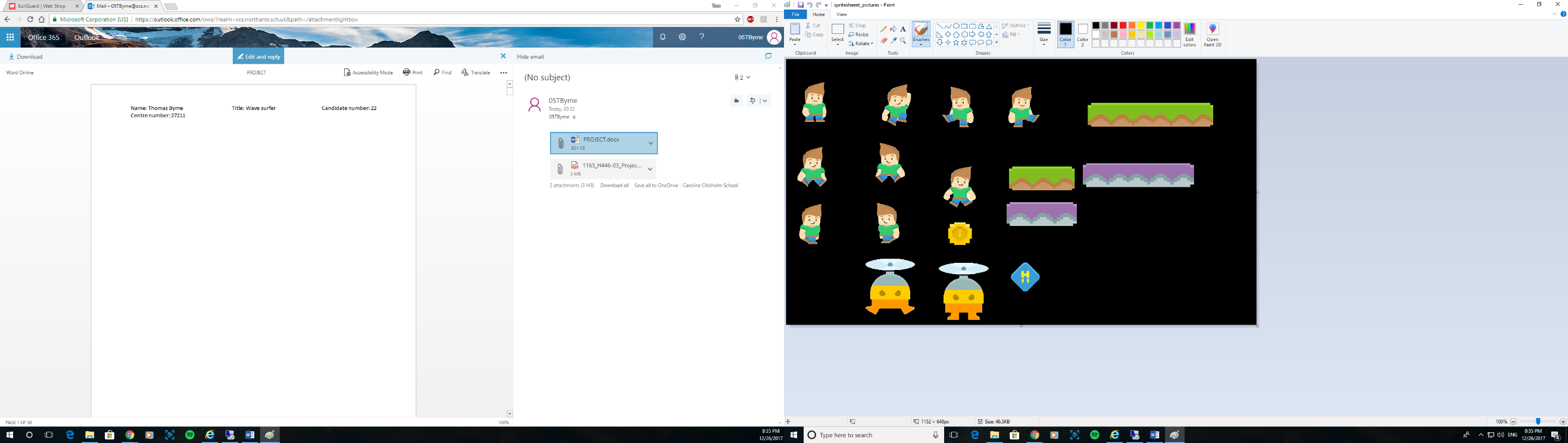
There are options to add comments throughout the code, I will be doing this to label what some classes are doing if I will be coming back to them regularly – this is so I can find them easily and I remember exactly what they do. They can be added by placing a “#” in front of the text I need to make into a comment.

When I add the sprite images for my platforms or character images, I can change their properties in my sprites file, when importing them I have to reference their coordinates and can decide on how big or small I want them in the sprite sheet or the code.

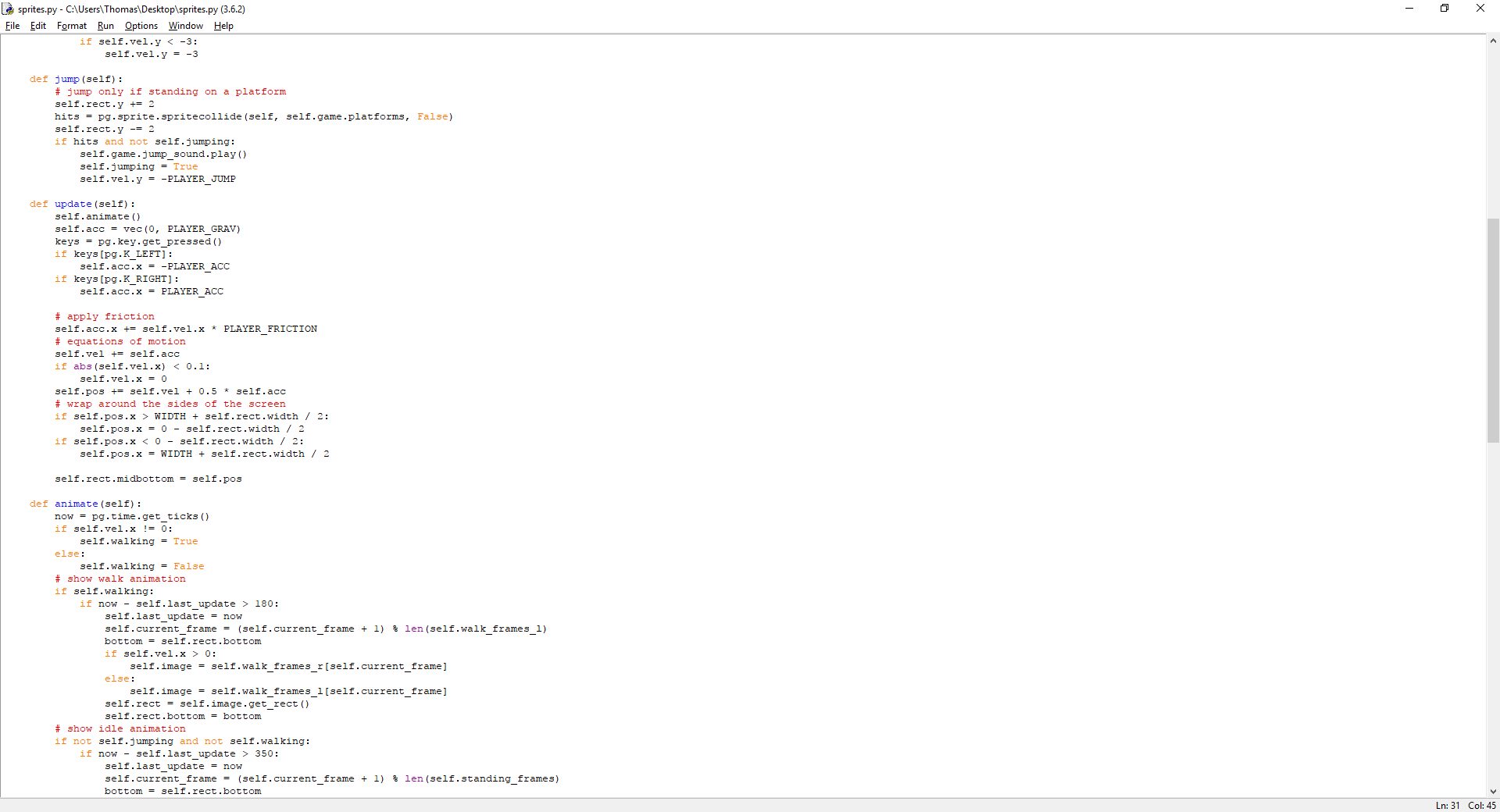


The number on the right (in red) specify the size of the spritesheet, whereas the numbers in the blue are coordinates for the game to find the image in the spritesheet.

The colour key in the code will be set to the same colour as the background of the spritesheet (Black), this removes any black from the background of my sprite and so only presents the image itself.

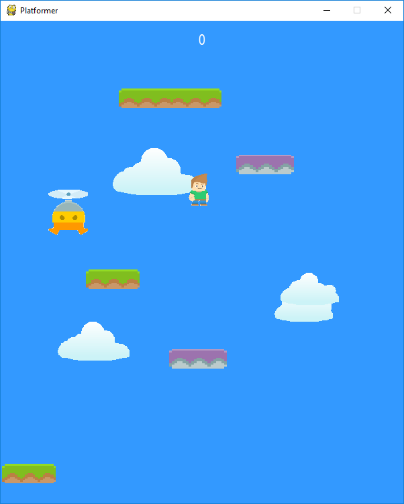
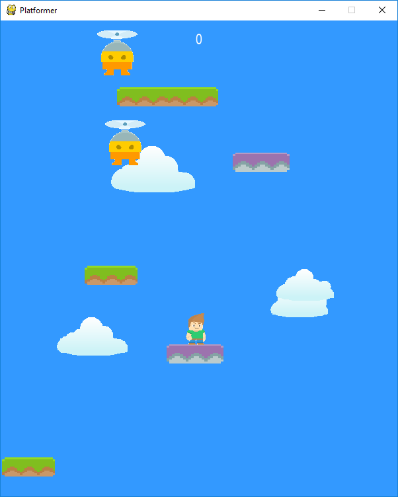
Sprite renderer and colliders

Each sprite in the spritesheet has set dimensions which make up the hit box, this allows the character to collide with sprites, such as 2D terrain, although it will not loose health by touching this terrain like it would if it hit an enemy, it allows it to walk across it, instead of sinking perpetually due to gravity. I added a black background so any details on the sprites that are white i.e. eyes it means they are not removed and do not fade in with the background. The sprites have no black on them which means they look smoother when presented in the game.

In my code I added a section that defined the position of the player, it was made to allow the player to travel across the screen and appear on the other side. By using the width and position of the player compared to the coordinates of the screen it could determinme when the player left the screen – how much left and how much to presnt on the other side.

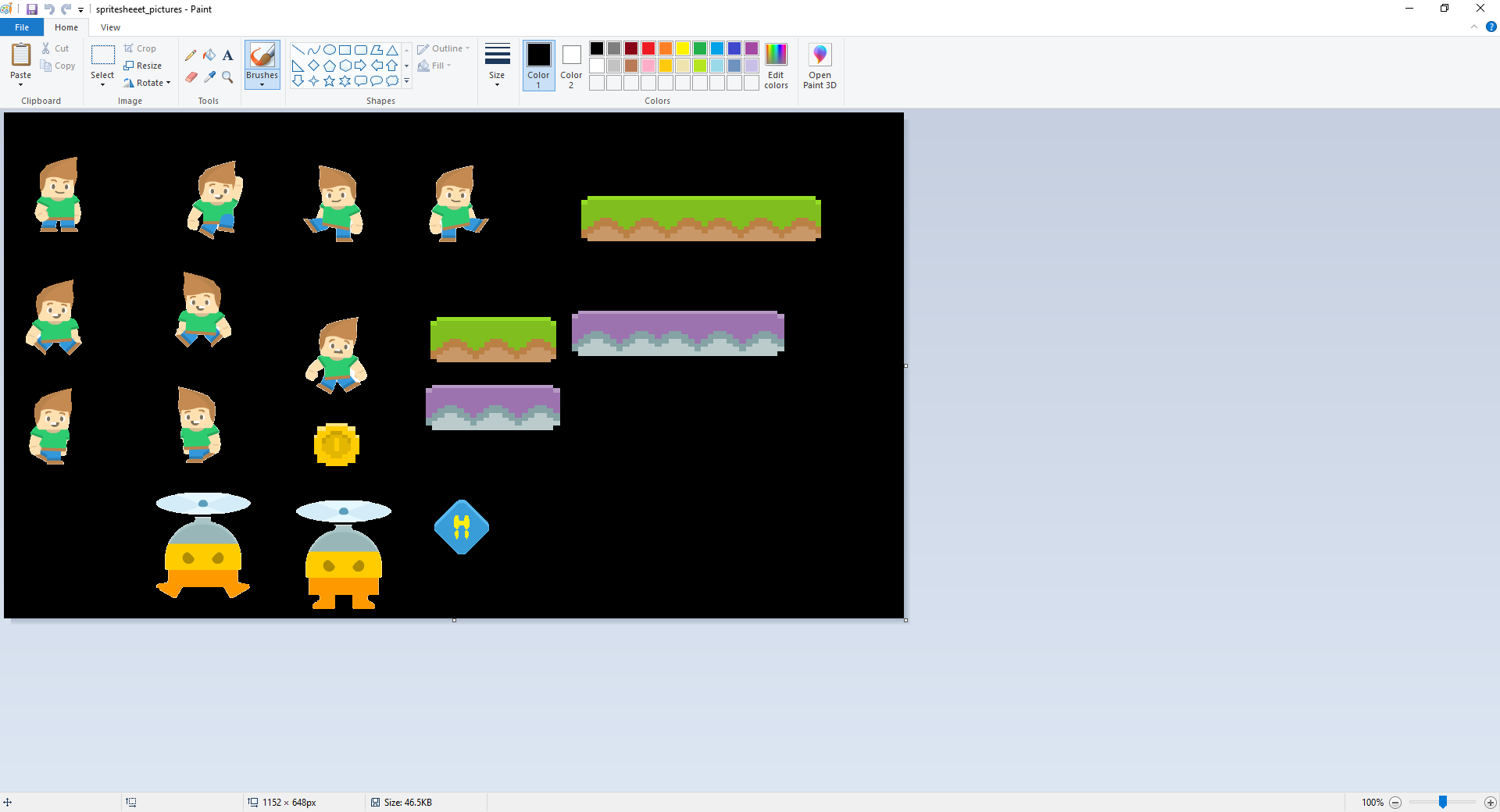
However all other sprites do not act like this, such as mobs or the 2D landscape, once the mobs leave the screen, they will be removed and a new one will be spawned in a random location. The 2D landcsape is made to stay on ther screen for cosmetic looks and practical uses, this means it makes the game slightly easier for ther player as they will be in easier to reach places/ closer to the character.

Earlier on in my code I made sure the player can not cheat by jumping in mid air, not only is this unrealsitic but can make ther game too easy for the player. It uses the players coordiantes of their feet and their velocity to determine whether they have collided with a platform yet. If their feet have hit it but their velocity is still > 0 then they are still decelerating. However if it is = 0 then it will mean they have landed and they are allowed to jump again.



Here is me testing my gravity and jumping to see if it is working, as you can see my charcater was it the air then my constant downwards velocity pushed him back down, emulating gravity. Also I was unable to jump again untill I had collided with that platform. The next step now is to add commands to move the player left and right, this should be pretty easy as I just need to add a loop, consatly checking for a key press that activates a velocity.

Idle animation

I need a few new character sprites, these will be shown in a certain order to make the movement seem more fluid/ realisitc. I will need to blit every frame to make sure they are usable.

There are bits of art I found online that can be sued to help with this movement, the character was looking a specific way in one of them, by flipping this it meant the charcter looked identical both ways. The software I sued to do this was just paint, it was a simple task and no ther features were needed, this kept the process simple and quick.

To create the animation I had to add a loop into the events section it the main part of my code. What this did was check if either the lft arrow or right arrow key had been pressed. If this = “= True” then the character acceleration (self.acceleration) = PLAYER\_ACCELERATION, this was defined in the settings earlier on as 0.5, the speed cannot be too high because then it would be too hard to control, and it cannot be too low due to the fact that then the user will not be able to land on the platform in time.

To make sure the changes in the frames are a smooth transition, I have added the images in by adding their coordinates and size, by providing two images it will flip between them both at set intervals, making it look like the charcater is walking by moving its legs.