**January 24th: Rapid Prototype start**

Given team Me, Tim, Matt C and Matt H started planning on who will do what on the rapid prototype. Using unity so we can have some idea of how systems will need to be designed and all of the team is very familiar with unity so there should be no problem finishing all objectives before the deadlines. Created a spreadsheet for all things that need to be done and split them into “musts” and “should” and “wants”. My first task was to get the Sprite simply moving and jumping with a platform for it to stand. I had to add in a grounding system so that the sprite knew when it was off the ground and could not continuously jump off into space. This was implemented simply using a Boolean and checking if the sprite is colliding with the floor sprite.

**January 28th:**

I have been set: adding a lives/death, creating a panning camera, points and a simplistic GUI to see where we might want to put info. Lives are just an integer that decrements if the player hits an enemy, to make sure the lives don’t go straight to zero because of constant collision with enemy the player is teleported back to a spawn point. Death happens if lives hit zero and the player stops being drawn. Panning camera was created by using “Sterilized field” with minimum and maximums set and then attaching the camera to the player movement, so the camera will follow the player until it hits the max or min. To record the score I set the amount of pickups collected to a string and printed it to the screen.

**February 7th:**

I have added a bit of smoothness to the movement which was previously a bit judder when players pressed multiple directions at once now it can only move in one direction per button press. I also added a small amount of drag so that the player slowly stops instead of it happening instantly. Most of rapid prototype work done now and peer marking done. Created a spreadsheet with the group for how we are going to hand out the work for the alpha. Next week will be spent getting to grips with the code and starting our assigned works.

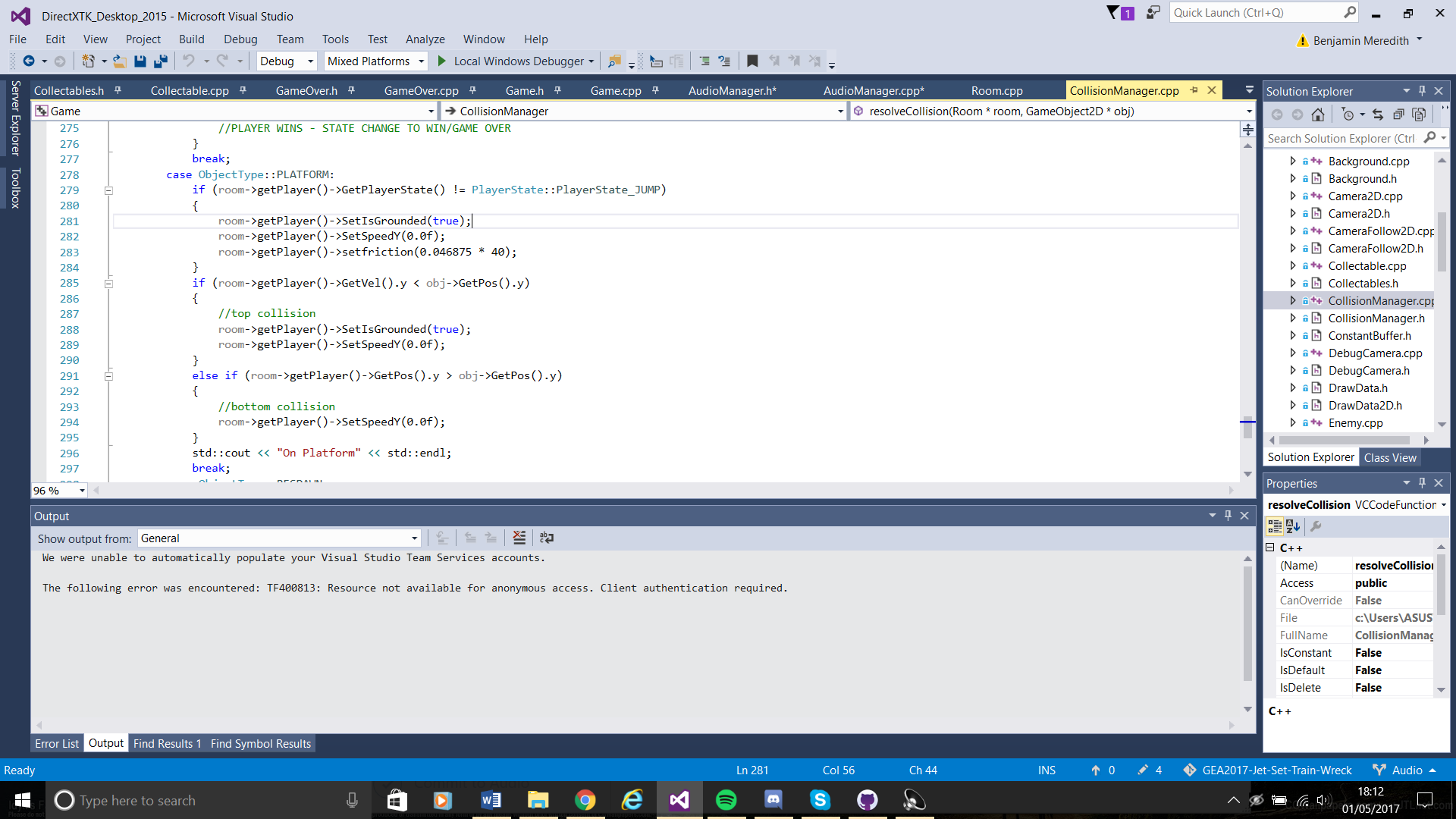
**February 14th:**

I have gotten used to the basics of the code and have started working on the collision system. I am using object types to define different collide able objects and allowing us to assign different rules to them in the future. This should make creating the jump through platforms much easier to implement. The collisions currently work by running through all objects and checking if they are at a certain distance from the player, if they are they are considered collided and movement is locked in the necessary direction. This will be changed to bounding boxes later but we will stick with this so that other work can be started.

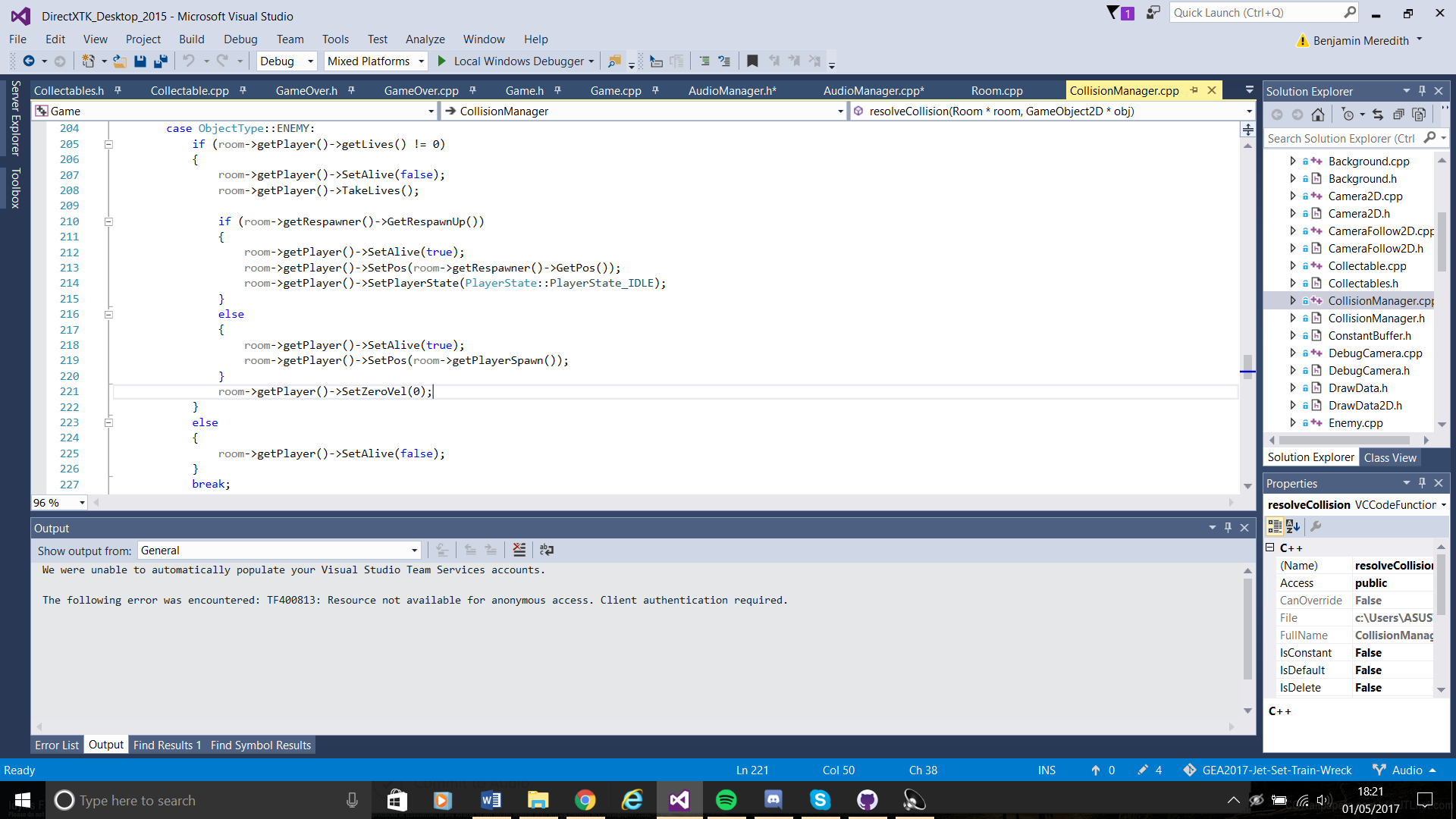
**February 21st:**

I have finished the Simplistic collisions system. Enemies and ladders have been added as well so I have moved to working on the deaths and lives for the player and a collection system. Now if the player collides with an enemy a life will be lost and the player will be teleported back to its original spawn point. As with the rapid prototype death happens when the players lives hit zero and the player stops drawing. Will be creating a respawn object for next week so when we create the level editor we can set player start points.

**February 28th:**

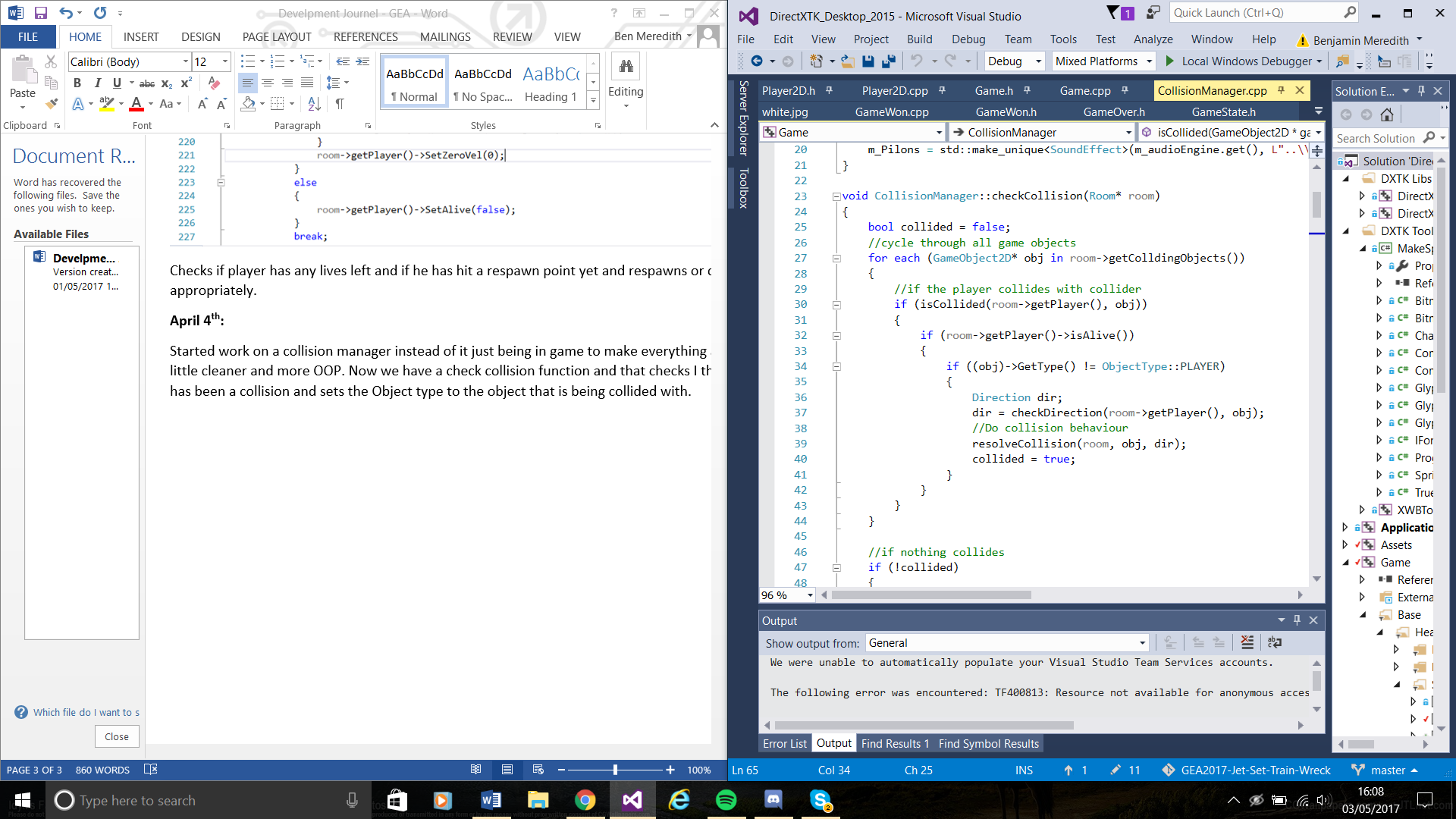
I have created a respawn point object as type collectable and added it to the collisions, so if the player tag collides with a collectable with object tag “RESPAWNER” then “SetRespawnUp” is set to true. Now if the player dies they will respawn at the position of this respawn object. If the player hits a different respawn object then then this information will be overridden and the player will respawn there. We decided to allow players to always respawn from the last respawn point touched instead of having them be sequential because will be doubling back on themselves, and this removes unnecessary walking. I also helped Cheung set up the platform and move through the bottom platform.

The Platform collisions decide whether the player is grounded or not and has friction to give the slow stop feel that we had in the rapid prototype. These collisions check each direction so we can use this code for the jump through the bottom platforms by changing the top collision section. I also helped with the enemy collisions so that I could implement lives and to link the player to the respawn point object.

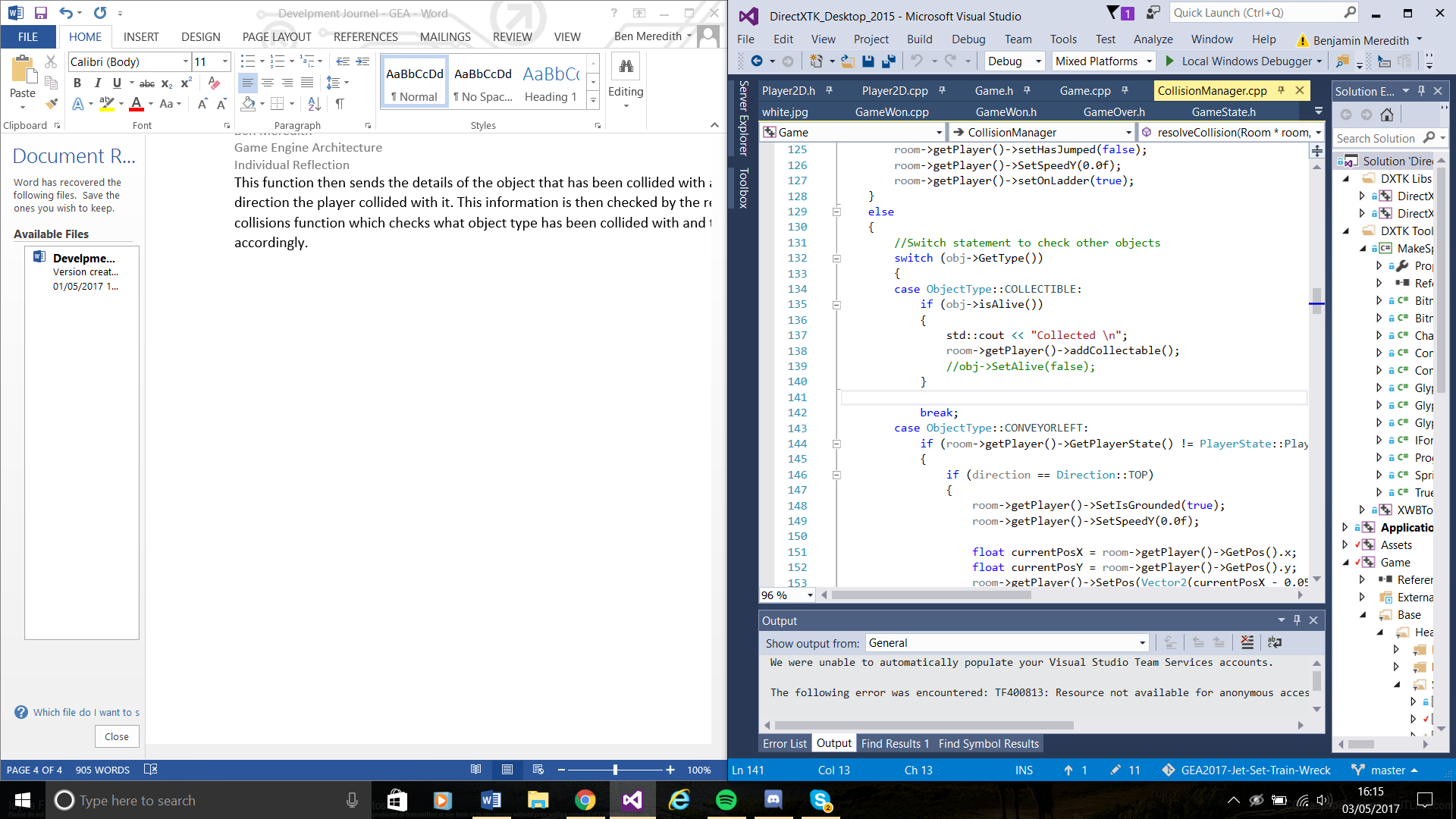


Checks if player has any lives left and if he has hit a respawn point yet and respawns or dies appropriately.

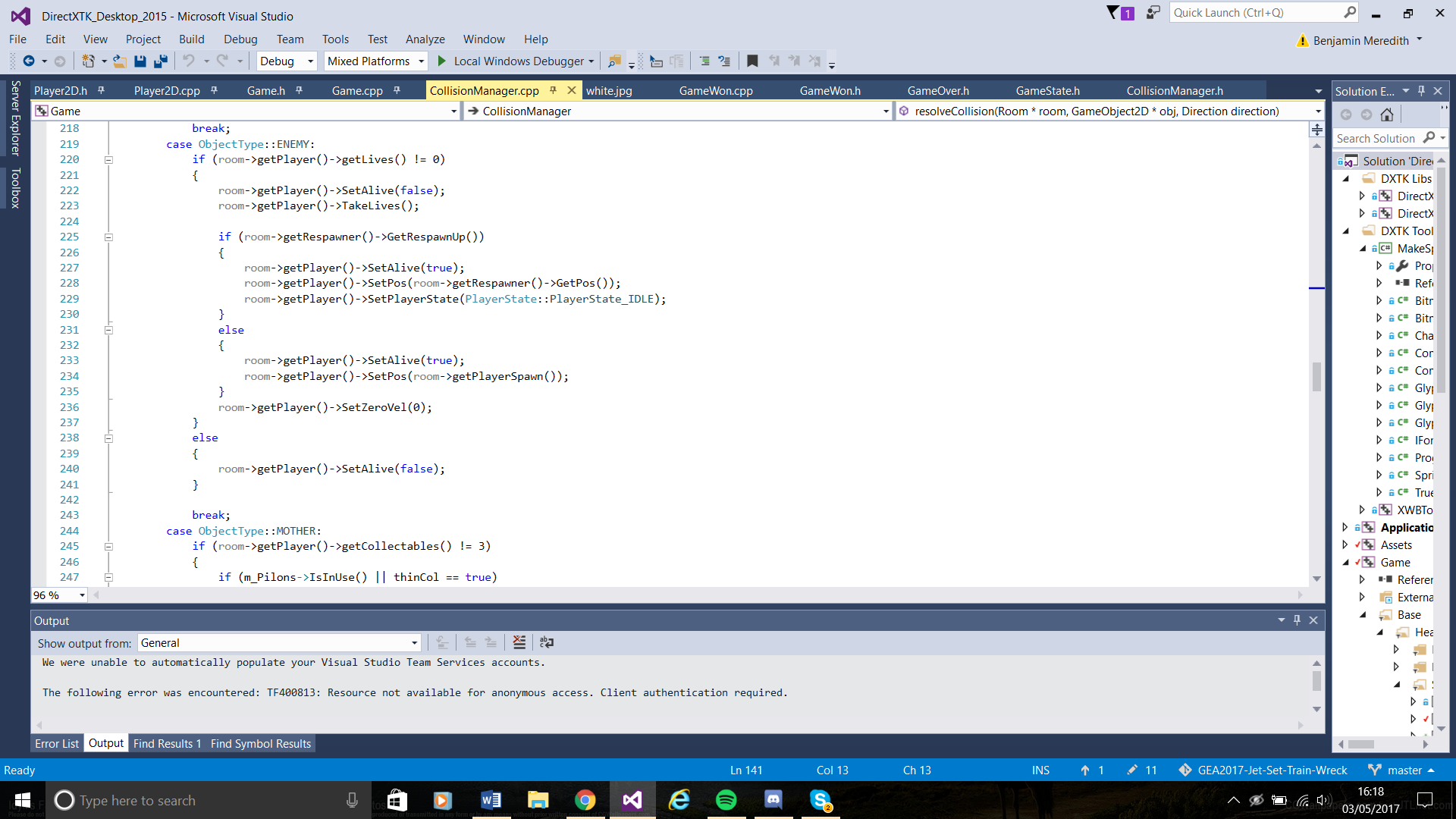
**April 4th:**

Started work on a collision manager instead of it just being in game to make everything a little cleaner and more OOP. Now we have a check collision function and that checks I there has been a collision and sets the Object type to the object that is being collided with.

This function then sends the details of the object that has been collided with and the direction the player collided with it. This information is then checked by the resolve collisions function which checks what object type has been collided with and then acts accordingly.



For collectables if a player collides with one then the collection number increments and the collectable is set to pick up/dead and stops being drawn.



If the player collides with an enemy Object then lives are checked, if zero then the player is set to dead the Game Overstate starts. If the player still has lives then respawn points are checked to see if the player has collided with any of them, if the player has then they will respawn at the position of the last respawn point touched.

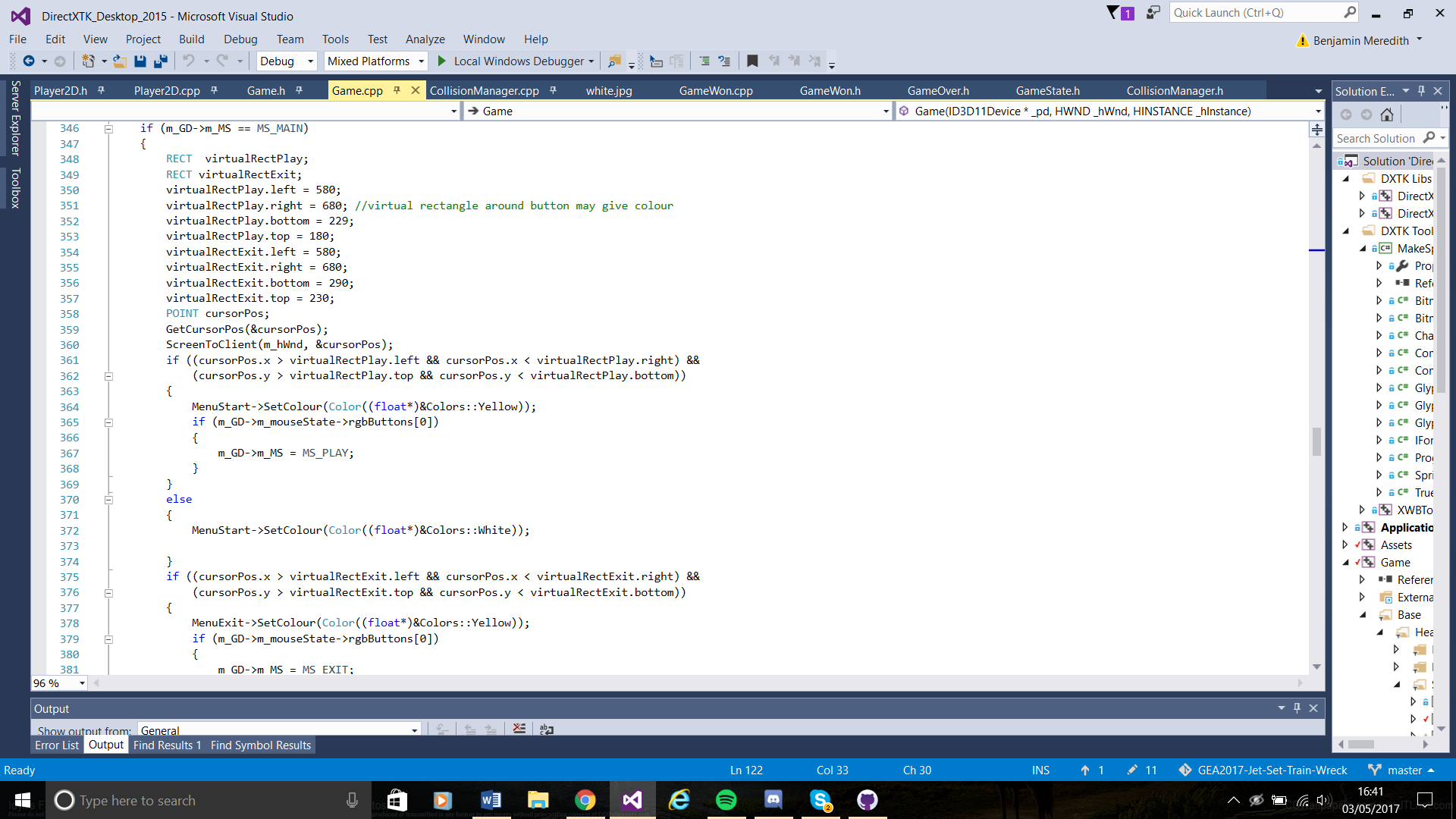
**April 9th – 23rd:**

Did not work on the project as I was focusing on Sim worlds and Low Level programming.

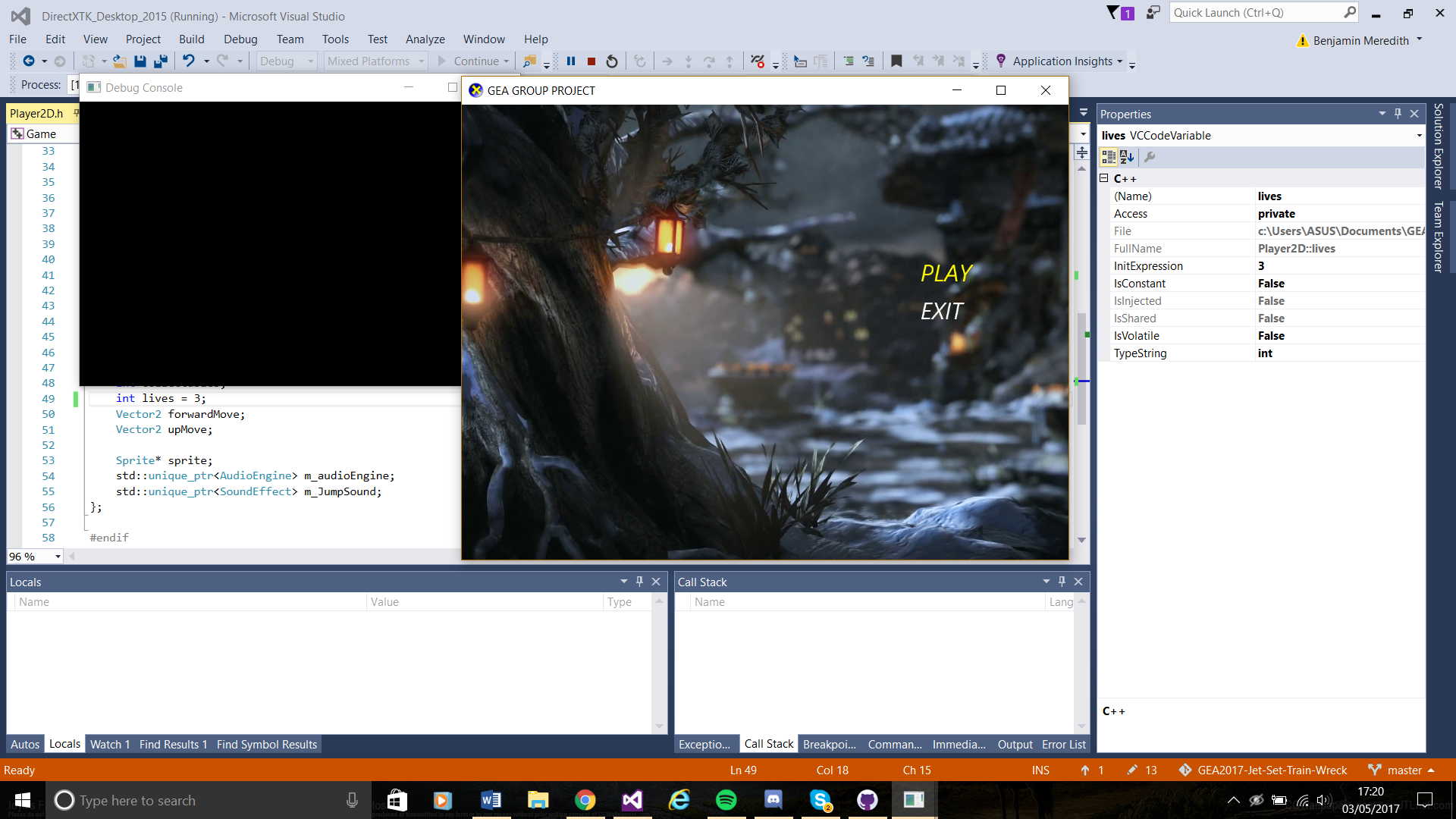
**April 24th:**

Fixed a bug with the player object which wasn’t loading in properly, it was caused by circular dependency, too many #includes. Also I fixed a bug with the respawn point where if the player died before colliding with a respawn point then they would spawn in position 0,0 which was in the middle of a wall. So I set the player to check where it first spawns and save that position so if it dies with no respawn point it will start back there. We are planning to create a respawn point on the position the player first spawns so this will be a temporary fix. Cheung and I spent a large portion of the rest of the day trying to fix some merge conflicts that had changed the file structure by putting everything in one file and had broken the physics. Eventually we moved all files back into their correct places and created a new repository to work from. Still have no idea why the files moved however.

**April 25th:**

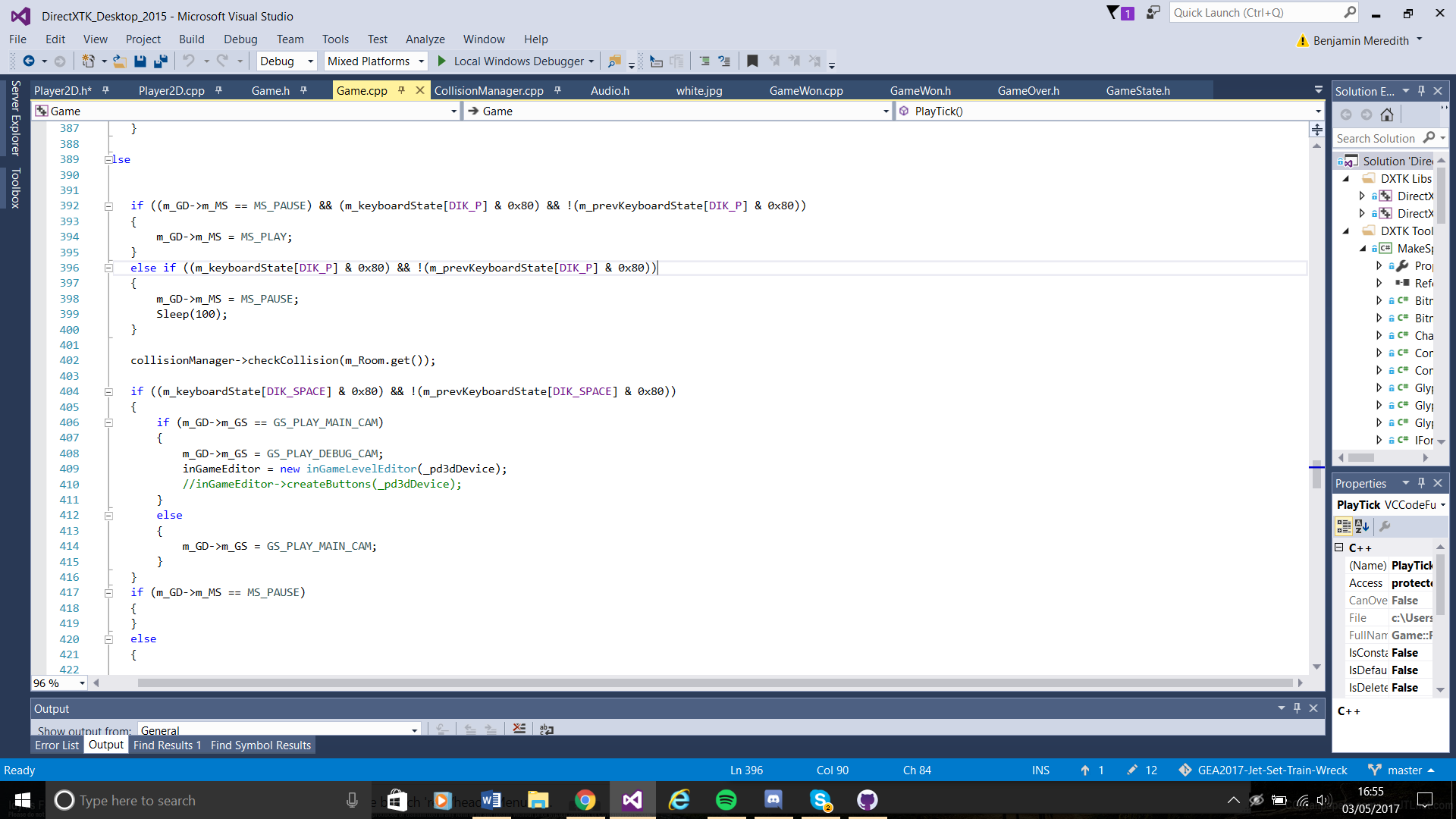
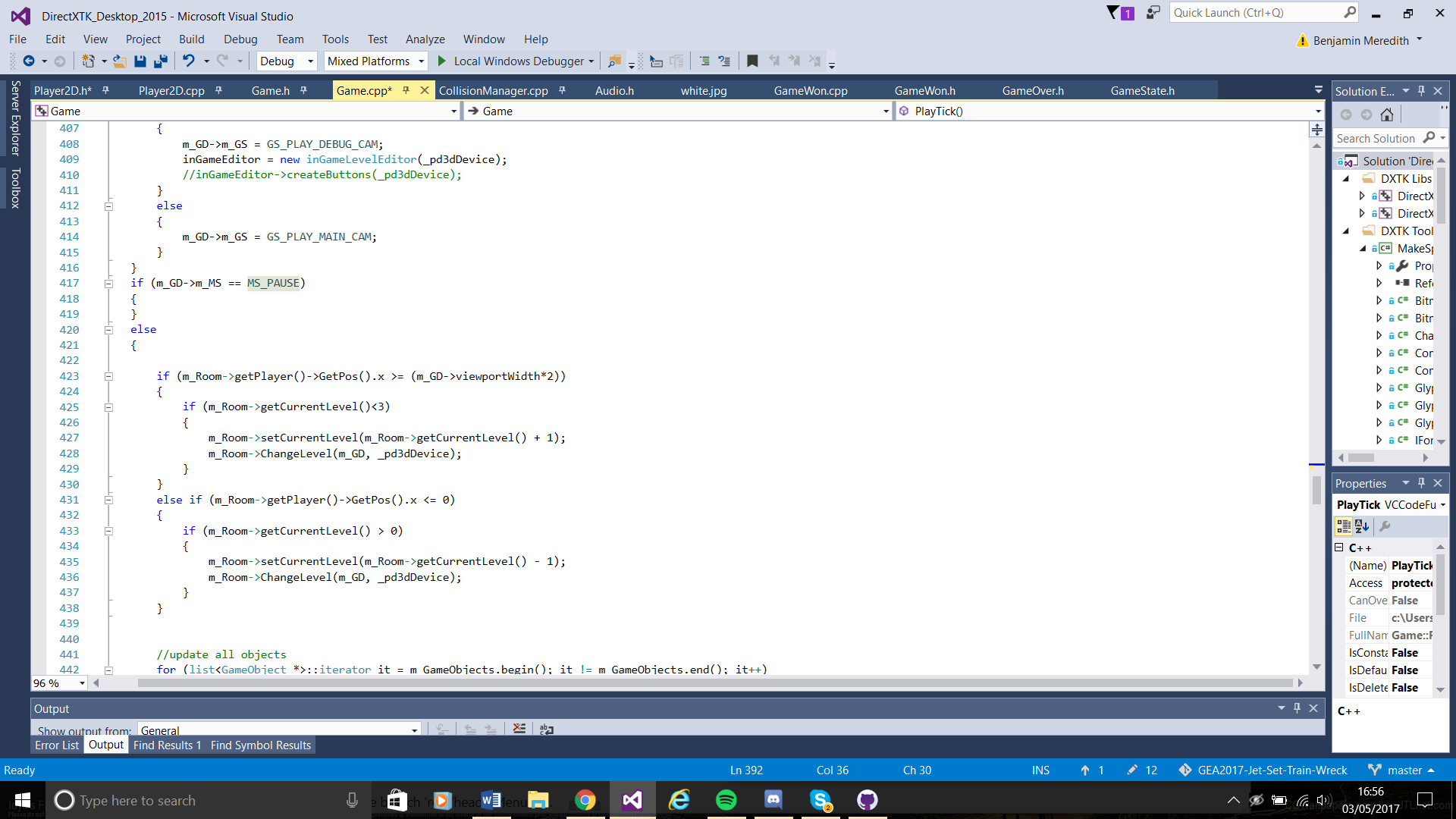
Finished making the main menu. Originally I was planning on making a scene manager that would change the scene depending on a game state but I was struggling to get it working. So instead I created a sprite that covered the screen and added the menu there, it is set to be drawn only if MS\_MAINMENU is active.

I created two RECT around the text to act as my button position. Using screen cursor I am also checking the position of the mouse and if the mouse clicks when on the Play button then the Play game State starts. If the player clicks exit then the game closes. I have also set the text to change colour when the mouse is hovering over it.

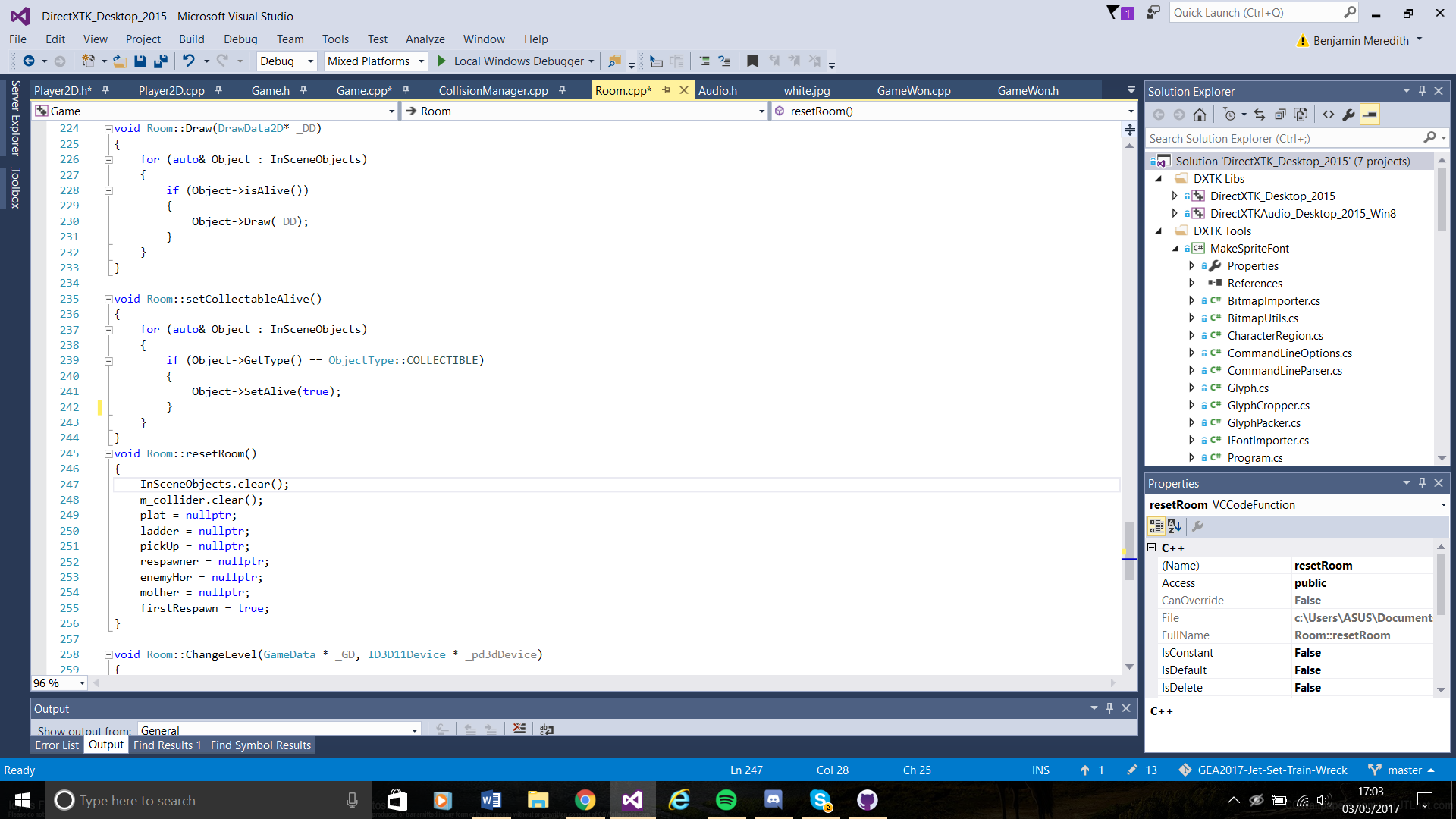


**April 26th:**

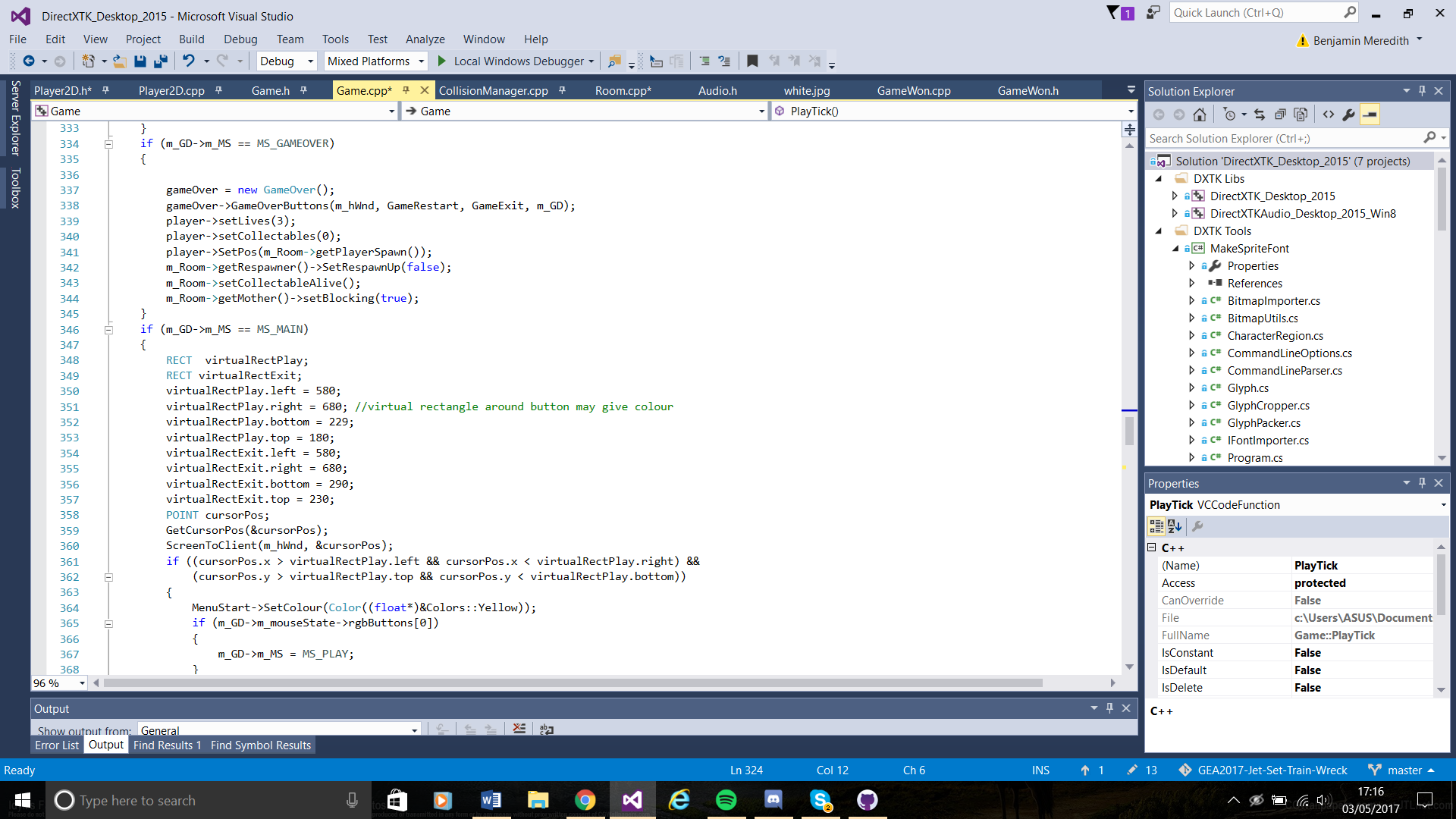
I have created a pause state. If the player presses P then MS\_PAUSE state is started and the game will not continue ticking until P has been pressed again. To stop the ticks detecting multiple presses of “P” when the player only presses it once I added a slight sleep on the press. Another way to solve this issue that I will do if time allows is using event handling instead and a case switch statement to check what is being pressed and it will only take the information once.



I also started work on a Game over screen with a restart button. The Game over screen works the same way as the menu but is only loaded when the Game Over state is active, caused when the player has zero lives and is in its own class instead of being in game.cpp. To create a restart button I had to set all the “dead” collectables to alive by iterating through the “InSceneObjects” which is a list containing all objects created in the scene.



The next part is setting the player lives back to 3 and their score back to zero which is done with two setters. This information is in player so I just create a pointer to player and call the necessary setters. Then I set the player spawn position back to their original spawn position and set all respawn points to false so the player wont respawn there until the player collides with them again. Then setting the Mother object to blocking. The mother object is the win condition collect all of the collectables and the mother object will stop blocking the player and the player wins.



**April 30th – May 2nd:**

Today we hit bug heaven. We changed the physics system slightly and added in more platform types but then collisions stopped working correctly so we spent a lot of time bug fixing. One of the bugs was we could climb up the left wall which we should collide with, it turned out it had been set to be a ladder game object and acted as a ladder but looked like a wall.

When that was finished I started work on the sound and gave sounds to jumping, background music and to when the player collides with the mother object. If the player does not have enough collectables then the game says “You must construct additional pilons”. I plan to add sounds to the button presses and a death sound as well. Originally I tried creating an audio manager by loading all audio during start up and then adding them to a list which could be called from anywhere in the program but I was struggling to get that working, and with the small amount of time left I wasn’t confident I could get it working properly so instead I used one shots and looping one shots for the music. It is a part of my work that does not look good but functions correctly and will be something I will come back and change after hand in so this does not happen again.

When creating the jump sound I had to make sure that the player was not using a ladder because if they were the sound would go off every tick and it sounded like a machine gun. So I checked for if the player is colliding with and ladder and limited the jump sound so it could only run one jump sound at once using the “isinuse”.

I then created the win state by adding to the menu state enum and checking to see if the player is colliding with the mother object. If the player is and the mother object is not blocking then the game ends and goes to the Game won screen which was made the same way as the other menu screens.

**May 3rd:**

The last bit of work was just some visual improvements, changing the sprites and the backgrounds of the different screens and a bit of code clean-up to make sure it fits code standard and is commented to a decent extent.

**Post Mortem**

Rapid Prototype

The Rapid prototype as a group went very well we made progress quickly and efficiently and had a lot of time to spare. As a group we worked well, there was little arguing and the problems we had were sorted out quickly. We were all very familiar with unity s found the work quite simple and if someone didn’t know how to do something the group would help that person to get the work done.

However we had a tendency to under reach. We focused too heavily on making sure that we completed all of the necessary parts of the game that we didn’t put much thought into extra features. This is the main problem that followed us through the creation of our project.

I personally feel that during the rapid prototype things went well for me. I was able to work with the group very easily. During the rapid prototype I chose some of the more complex work because I was confident with Unity so I worked on the rope and camera settings as well as the state machine. However I also did not try and put the extra effort in and add cool features, instead we put them in a list of maybes which we never ended up completing.

**Alpha**

During the Alpha the group was still working well as a cohesive unit, we had made a spreadsheet of all jobs needed to do and their time constraints and everyone knew what they were doing. We had a few disputes about code standard and layout but they were quickly resolved. As a group we were working very well. Apart from our problem of creating things that just worked instead trying to make them interesting to use.

During the Alpha I worked well overall, I created the collisions system as fast as possible and set it up to be easily editable so that everyone could start the other parts of work. I became used to the codebase quickly but was not willing to move away from Simons work that I was more used too. But my ability to work with DirectX has definitely increased and I enjoy working with it a lot more than I used too.

**Final Submission**

The entire group spent about a month of not working on this project and working on other things. This caused some problems, it took us a few more days to remember what we were doing and get back up to speed but because we all did it we were able to help each other get back into the swing of things. But it was at this point we started noticing that our management and ordering of things wasn’t the best. We had delegated out work but we realised certain things need more attention than we thought and we hadn’t set anyone to plan ahead and time manage. Usually I try and take that role when in groups but I didn’t in this and Matt Holmes took up the role. For next time we need to choose a person to manage the team and delegate out jobs as well as keep track of time constraints.

Personally during this I believe I didn’t take on enough challenging jobs and I should have attempted to finish my small parts quickly and then start adding extra features to make the game more interesting to play. Because of this our end result is a working game but lacking in extra features. This all revolves around my current problem of only striving to do what is necessary and not exceed expectations. That is a part of my work that definitely needs work. I also started getting lazy with my coding the closer to the deadline I got, instead of doing things the harder but more correct way I chose to do things simply and easily so they would work. This can be shown from my work on the audio and menu systems. For next time I will attempt to manage my time better so that I don’t get into the situation where im worried about not completing work and just botch it together.

Overall as a group we worked very well and have produced what was asked for but not a lot more. We all need to improve our management and try and strive for more than just the necessary. I need to improve my code standard to make it more consistent and lay it out better to make it more usable if someone else needed to use it. But all in all this project went quite well and I have found good points to improve on for next time.