|  |  |  |  |
| --- | --- | --- | --- |
| Name: | Tia Mulholland | P Number: | P2433472 |
| Github Username: | Asia292 | Github Repo URL: | https://github.com/DMU-IMAT2906/cw-Asia292 |

**Please summarise the functionality of your platformer**

* Player moves left, right and can jump
* IsGrounded bool which controls whether player can jump doesn’t always change to true on collision with chain shapes
* Enemy patrols between defined coordinates
* Enemy stops patrolling and goes squish when player jumps on it once
* Enemy dies (sprite & b2Body deleted) when player jumps on it twice within 5 seconds
* Enemy restarts patrol if player doesn’t jump on it a second time in 5 seconds
* When jumping on bee enemy it falls underneath the player instead of staying in place
* Player rebounds and enters “daze” state when hits enemies side or when collides with a hazard
* Bouncy platforms add a high upwards force to player
* Moving platforms move between two defined coordinates
* One-way platforms allow player to pass through while underneath
* One-way platforms allow player to stand on them when above
* Door slides open and closed using a motor joint and controlled by sensor
* While sensor is pressed door opens/stays open
* While sensor isn’t pressed door closes/stays closed
* Item deleted (sprite & b2Body) when picked up by player
* Score is updated on HUD depending on what item is picked up
* Time decreases on HUD each second
* Life sprite removed on collision with enemy/hazard
* If lives = 0 then game stops updating, player “poof” animation plays and game over screen displayed
* Coin requirement on HUD updated on coin pick up, says ‘EXIT’ if requirement reached
* Sensor checks to see if the player has reached home
* If player has reached home and coin requirement reached, game stops updating and game won screen appears with score plus time bonus
* Animation frames advance and loop if required
* Animation speeds vary
* Player and enemies change animation based on actions

**What testing have you performed and what testing strategy was used?**

I regularly outputted statements to the console when implementing a new functionality, this helped me see if the statements where being run and what values where being changed. I also originally had placeholder statements within my contact listener to test if the collisions where triggering properly before I had implemented the desired affects. The two main areas I used cout was when setting up the contact listener and checking that the enemies hit and die/return to patrol worked. For the contact listener it was comparing the user data it was receiving against that of the object (I originally wasn’t dereferencing the pointer) and seeing what was being changed on contact. In terms of the enemy, it was used to see what bools where being changed and passed between the contact listener, the enemy itself and the game. I had an issue where hit once was being reset upon return to patrol but hit two was what was triggered next time. This was resolved by resetting both bools on return.

Alongside these statements I also ran several Google Unit Tests. These where used to confirm that values where changing and that they changed as expected. The tests were split up into 8 test suites – singleton, animation, player functions, enemy functions, platforms, door sensor, home sensor and HUD updates. The singleton tested to see if the texture manager was probably implemented as a singleton. Animation focuses on how the frames advance based on different variables. The player controls are tested in player functions with enemy functions focusing on the patrol/hit feature. The one-way platform system is tested along with moving platforms direction changing in platforms. Door and home sensors check their on/off action functions respectively and HUD updates makes sure values are being changed on respective collisions.

I also ran two performance profiles; one where the player died half-way through and one where the game was completed. Both showed similar data with drawing and updating taking up the highest percentage of the CPU usage as expected. Other than this everything had low usage, so no major changes needed to me made to increase its performance.

**Are there any parts of your work that you are particularly proud of?**

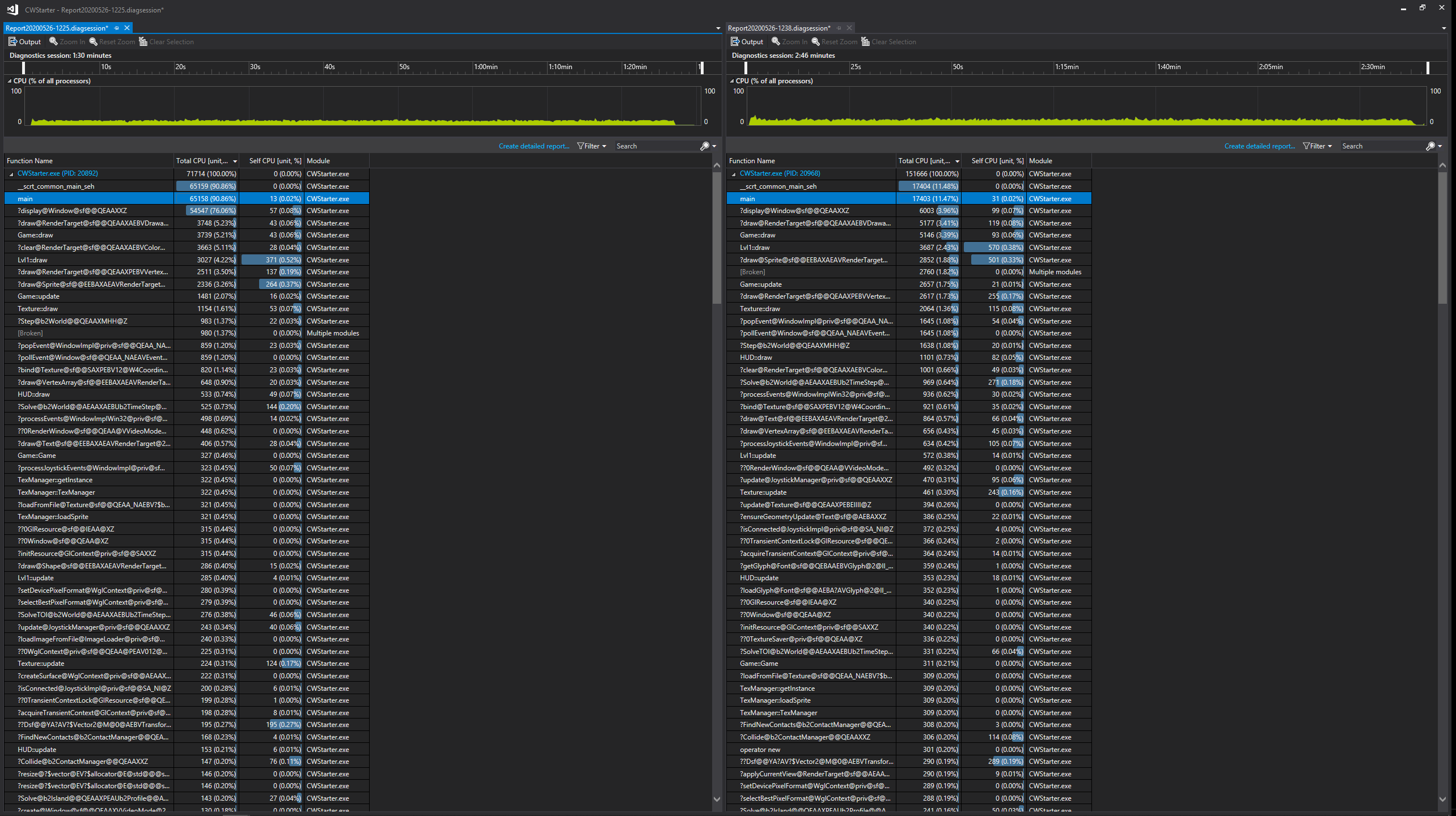
**Texture manager** – Implemented as a singleton, it uses polymorphism to apply changes to classes inherited from Texture. It also stores a pointer to all the sprite sheets as well as vectors of all the key frames intRects. It combines two techniques I have never used before and they are both implemented successfully.

**One-way platform** – I had several ideas of how to implement this ranging from moving the player past the platform, disabling and enabling the platform entirely or doing the same for the collision. I was unable to implement any of these ideas - I couldn’t reliably alter the players position, in order to disable a b2Body I would have to delete it and remake it (something that couldn’t be done in time) and completely disabling the contact listener was a bad idea. In the end I decided to use collision filtering on the platform itself. It has two functions – collide and pass – which set the body’s filter data accordingly. A check is run in update to see if the player is below the platforms position – in which case pass is called which sets its mask bits to 0x0000 – or if it is above – collide is called and its mask bits are set to 0x0001.

**If you were to undertake this piece of work again what would you do differently?**

Due to time constraints, I wasn’t able to implement everything I had originally planned. I had planned to have at least two levels showing different game mechanics for the first level (the one I implemented) is just an introduction to everything within the game. The idea of how to do this was each level would have its own class inherited from a base one. Game would then have a level pointer to the current level (not one specific to a level like it does currently) which by default would be level one. On level complete (dictated by a bool) it would destroy the current level and the pointer would be set to the next one and the bool reset to false. Due to HUD being only linked to game, this would carry over into the next level.

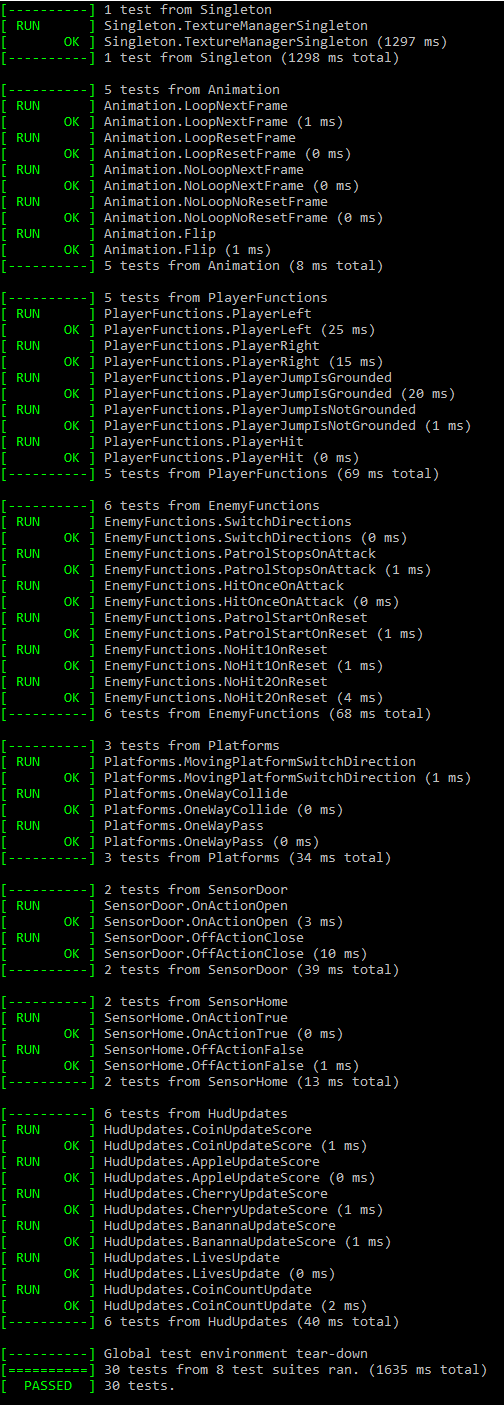
Another thing I would do is separate the enemy and pick up classes. Due to both types of enemies being in the same class, to make the bee stay in the air I set its gravity scale to 0. This has the draw back however of it falling when the player jumps on it to kill it. For pick-ups, it would be more practical rather than defining its sprite and animation properties using a switch case based on a char passed in the constructor.



Performance Profiling output

Full game

Half of game



Google Unit Test output