**Introduction**

This project is being developed on Unity thus each testing (Unit, Component, System) can be conveniently done by playing the scene instantly, each time new block of code affecting the scene is created. With that being said, I will test each method or class by simply playing the scene every time I add new behavior/object script to the game. Particularly, scripts (C#) written inside Unity have method called Start(), Update(), OnCollisionEnter(), and many other method to implement behavior/characteristics to the game object. In Start() method, variables are simply initialized thus I will not talk about it in the testing section. Update() is one of the most crucial methods in the game project. It gets called 60 times a frame so it is generally used to make some object to do some action. I will mostly talk about this method in the testing section. OnCollision() method is called when there is an interaction between two different object, so it will be mostly tested in the component testing section. Many other custom methods than those method mentioned above will be tested by actually playing the scene.

**Unit**

[1. ThePlayer.cs]

* Update() : This is where I receive the user input from the keyboard to control the player character object in the scene. By playing the boss stage scene, I tested if the player sprite responds and move correctly by the user input. The left and right arrow keys on the keyboard should make the object to move along horizontal axis, and the ‘c’ key to make it jump, ‘z’ and ‘v’ key to transfer the object in certain distance along the x-axis and y-axis respectively. By shooting invisible ray downward in unit distance, by using Physics2D.Raycast(), to check if the object is in contact with the platform object so it plays the appropriate animation correspondingly. Also, this is where to track the players hp variable value continuously, and if the hp value becomes less or equal to zero, Update() stops receiving user input, meaning the object has no more movement in the scene. It was check that the player object moved properly according to the user inputs, and made sure that it stopped responding to the user input as its hp value reaches 0. In addition, I checked if unassigned key inputs trigger any unexpected behavior of the object, and fixed it if there was any.
* Flip() : It flips the player character’s sprite as the character changes its direction of the velocity vector thus it naturally displays the change in direction to the user. The game sprite was properly flipped based on its direction as I tested by controlling the character with arrow keys. I also tested if there was any unexpected Flip() call by interacting with other game object, and fixed it if there was any.
* TakeDamage() : This is the method where it receives the monster attack prefab’s damage variable value and passes it to the player’s hp value to reduce it. Test was done by getting hit by the monster attack prefab by playing the boss stage scene. I made sure that the player object hp variable goes down with proper amount according to the monster attack prefab, and was tested for every single monster attack prefab to check if the method works fine.

[2. Weapon.cs]

* Update() : Inside this method, the script is ready to receive ‘x’ key input from the user to call Shoot() method. Also it plays different audio source based on the player character’s phase because the character shoots different bullet on different phase. Appropriate sound was played based on the right phase, and Shoot() method was correctly called by the input.
* Shoot(): This is where to instantiate the attack prefab. Two different bullet prefab are instantiated here according to the player character’s current phase. It checks the phase by checking the animation state of the player object using animator.GetBool([animation name]). It was tested by playing the scene, pressing ‘x’ key on different phases. Correct prefabs were instantiated based on the phases, and it was tested for every animation of the object (walking, jumping, and IDLE standing).

[3. TheMonster.cs]

* Update() and FixedUpdate() : There are two phases for the monster ai object, the first and the second. The object starts with the first phase, and the second phase starts when the hp value goes below 1000. Each phase plays different animation thus this method should keep tracking its object’s current animation state name. When CurrentAnimatorStateInfo(“1st phase”) is true, the object’s rigid body gravity scale is active so the object falls when there is no platform object is below. When CurrentAnimatorStateInfo(“2nd pahse”) is true, the method sets the gravity scale equals to zero so the object is free in vector space. Under both phases, the method keeps tracking the local position of the player object, and move the monster object to where the player is. Each phase was tested to see if the monster ai properly locate the player object and follows the path. To check every possible outcomes, I controlled the player object to be in various positions and I confirmed the monster ai correctly responded to that. As I intended, during the 1st phase, monster could only track and follow the player along the horizontal axis since it’s bounded to the platform due to the gravity. On the other hand, the ai during the 2nd phase, could track and follow the player along the both horizontal and vertical axis as it’s free in gravity factor. As the hp value of the monster goes below 0, the method immediately set the vector velocity value of its object, thus it can still track but can’t follow the player, meaning no more movement.
* TakeDamage() : This method was tested in the same manner as the same named method in ThePlayer.cs script since it has the same purpose of receiving the damage value from the player’s attack prefab. I controlled the player character and attacked the monster ai and confirmed the monster’s hp value drops properly. In addition, this method also invokes Die() and DieAndScream() method when the hp value drops below 0, to bring the death effect of the monster object in the scene. Those methods were properly called and checked by keep attacking the monster ai until the hp becomes 0. I also modify bullet prefab’s damage value to be much greater to test the method if it works properly under the dramatic circumstances.
* Flip(): The same purpose as the same named method in ThePlayer.cs and tested with the same strategy. Tests were positive as it was intended to be.
* Die(): It calls Destroy(gameObject) to destroy the object the script is attached to. It was tested by killing the monster object and check if the object is actually removed from the hierarchy window on Unity.
* DieAndScream() : It simply plays audio source upon calling. Audio was properly played upon the method call.

[4. Skills.cs]

* Update() : I am going to use recursive invoke methods to make the monster ai to keep instantiating the attack prefabs in some intervals. In Update() method, there is only simple block of code to check hp value in TheMonster.cs if it’s below 1000, and if it is, then I set Boolean variable true to invoke skill method for second phase of the monster ai. I tested if the attack prefabs in second phase were launched properly as the hp value goes down below 1000, and it worked fine.
* SkillChoices\_phaseOne() : This method instantiates four different attack prefabs for phase one, and the attack to be instantiated is selected by using switch with random integer variable ranged (0, 4). Instantiation of each attack prefab will be called inside other methods that will be discussed below, and switch just randomly invokes one of the methods. When the Boolean variable checking if the monster is in second phase becomes true, instead of recursively invoking this method, it cancel the current method, and invokes SkillChoices\_phaseTwo() to recursively instantiate attack prefabs for the second phase. To test if this method works properly, I simply played game scene and spectated the monster ai if it launches all four attack prefabs randomly. I also commented out three switch cases leaving only one case to test each attack prefab individually. Lastly, I attacked the monster ai to drop its hp below 1000 to see the monster stopping invoking this method and invoke SkillChoices\_phaseTwo() properly.
* SkillChoices\_phaseTwo(): This method has same concept as the method right above, randomly invoking skill methods to instantiate attack prefabs using switch case with random integer variable for the cases. Test was done in the same manner as testing the previous method, and confirmed all the attack prefabs were launched randomly.
* ShootBullet(), ShootSkullBomb(), ShootAxe(), ShootAxe(), EyeBlaster(), BloodShowerOne(), BloodShowerTwo(), BrainBlaster() : those are the simple methods that instantiate corresponding attack prefabs that will be called inside the switch-case. Testing was done while testing above two methods, confirming that the attack prefabs were instantiated at the proper given firepoints’ vector positions.

[5. CameraBehavior.cs]

* Update() : It checks the Boolean variable named isShakeTriggered and if it’s true, it randomly resets the camera position inside the unit sphere centered in the initial camera position for given period of time to deliver shaking effect to the player. Timer.cs sets isShakeTriggered true when remaining time variable inside Timer.cs becomes zero. To test the method, I changed the time limit from five minutes to five seconds to efficiently check validity of the method at remaining time = 0.
* LateUpdate() : This method lets the camera object, which this script is attached to, track and follow the player character’s local position. However, the range of its transform is limited to the scene size thus the camera accidently shows undeveloped scene space to the user. I tested the method by controlling the player character to move to each corner of the scene and edges to check if the camera is correctly bound to the limited range, and if it properly follows the player character’s path.

[6. Timer.cs]

* Update() : This method is supposed to decreases the starting time as the time flows, and stores it in remaining time variable until the remaining time becomes zero while the monster ai is alive in the scene. Remaining time should be displayed in the screen thus the player knows how much of time is left to defeat the monster ai. Remaining time variable contains time value in second, and the code in this method should convert second to minutes : seconds. I tested and confirmed that timer is properly displayed in the screen by playing the scene with given initial time (5 minutes). Timer.cs script interacts with many other scripts attached to game objects/scenes, and testing their interactions will be discussed in Component testing section.

[7. Scripts attached to Attack prefabs]

BrainBlaster.cs, BloodShower.cs, Eyeblaster.cs, EyeBullet.cs, Hellfire.cs, SkullBomb.cs, SpinningAxe.cs, Bullet.cs, BulletSniper.cs are the scripts that are attached to the attack prefabs of player/monster. They all have simple methods that sets initial damage value and velocity and pass the damage value to the opponent. Their validities are tested while testing Weapon.cs and Skills.cs which are the scripts that instantiate those prefabs either automatically by algorithm (monster ai case), or by receiving user input to trigger it (player case). Unexpected situation such as wrong travel direction/velocity, not passing the damage value to the opponent, incorrect sprite size and others have been fixed by constantly playing the scene and checking the validity of each.

**Component**

|  |  |  |
| --- | --- | --- |
| **Scenarios** | **Interactions** | **Tests** |
| Interactions between Game objects and Prefabs | Interaction between Monster/Player object and attack prefabs | a) When player/monster object with a tag “Player”/”Monster” interacts with attack prefabs tagged “Monster Attack”/”Player Attack,” hp variable inside script attached to player/monster object should be reduced by proper damage value passed from the prefabs respectively.  b) Test if any unintended interaction between “Monster” and “Monster Attack” or “Player” and “Player Attack” cause unwanted damage passing method calls |
| Collisions between attack prefabs, player/monster objects, and platform | a) Monster and Player objects should physically collide with platform layer thus the platform can work as a ground for those objects. Play the scene and check if the objects don’t fall passing the platform.  b) Test if unwanted collision between monster object and player object (they are layered as “Monster” and “Player” respectively) occurs. Play the scene and control the player character moving toward the monster and see if a collision occurs. |
| Interactions between Game objects and Game Scenes | Interaction between Player object and Game over scene | a) Test if the method named Gameover() gets called properly from ThePlayer.cs when hp variable becomes zero, and loads Game over scene. Controlled the player character to get hit by monster attack prefabs until its hp value became zero and confirmed that Game over scene get loaded.  b) Test if there is any unexpected point triggering game over scene by controlling the player character dodging all “monster” prefabs until timer becomes 0 and that loads up Game over scene. It has been tested and confirmed. |
| Interaction between Monster object and You win scene | a) Test if hp variable inside TheMonster.cs becoming zero triggers a method named OnDestroy() to load You win scene. Control the player character to attack the monster ai until its hp becomes zero and confirmed that You win scene is loaded afterward.  b) What happens when hp of monster and player both become zero simultaneously? Resolving such abstract problem by setting the player/monster object layer “Invulnerable”, which is unaffectable layer by all other layer, immediately when the opponent object becomes zero hp. |
| Interactions between Game scenes and Game scenes | Interaction between scripts attached to a scene and another scene | a) Each game scene has a script to load other scene upon user request. Each script has simple method loading corresponding scene and the method is called upon clicking a button which references the script. Tested all the buttons oriented in each scene and confirmed that corresponding scenes were properly loaded by clicking the right button. |
| Interactions between Timer and Game objects/scenes | Interaction between timer and Monster object | a) Update() method in Timer.cs script references TheMonster.cs to track Monster’s hp value and when the hp becomes below zero, Update() stops updating remaining time variable, thus displayed timer on the boss fight scene gets paused. Then, OnDestory() method inside TheMonster.cs references Timer.cs and gets passed remaining time value and stores it in minutes : seconds (string) format in a file in the local folder (it will be used for the score board). Tested that the timer stops upon monster’s hp becomes the zero and confirmed that score board displays correct record from the last play.  b) What happens when The monster hp becomes zero exactly when the timer also becomes 0? Resolving the issue by changing Update() method in Timer.cs to LateUpdate() which comes one frame after Update() method. By the resolution, even if the monster’s hp becomes zero, loading Game over scene as timer becomes zero will be executed one frame later than the monster death loading You win scene. |
| Interaction between timer and Player object | a) When remaining time value becomes equal or less than zero, the player object will stop responding to the user inputs, and LateUpdate() method inside Timer.cs will load Game over scene. To efficiently test the interaction, I temporarily set the start time as 5f (5 seconds), and confirmed that I lost the control over player character as the timer displays 0:0 on the screen. |

**System**

After testing and confirming that all units and components of the project work properly as intended, I, as a player, played the game from the very beginning. I have already had too many experiences to this game as developing the game, thus I considered myself as an “expert” level player to this game. I wanted “expert” level player takes about 3-4 minutes to clear the game, so I modified few small factors (player damage, monster attacks’ damage, Monster ai invoking each skill interval, velocity of monster ai) to fit the level I desired as I keep playing this game for 15-20 times. After I determined the appropriate level for the game, I let my friends, who occasionally enjoy playing similar type of games. I assumed them as a “beginner” level players to this game, and I wanted them to struggle defeating the game for at least 5 trials. I asked three friends to play the game for 7 trials each and the result is following:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | 1st | 2nd | 3rd | 4th | 5th | 6th | 7th |
| Heeyoung Lee | Lose(0m57s) | Lose(0m47s) | Lose(0m55s) | Lose(1m25s) | Lose(1m42s) | Lose(1m2s) | Lose(2m33s) |
| Anderson Yoon | Lose(0m33s) | Lose(1m33s) | Lose(2m2s) | Win(3m43s) | Lose(1m3s) | Lose(2m2s) | Win(3m12s) |
| Daniel Yoon | Lose(1m22s) | Lose(1m30s) | Lose(2m10s) | Lose(1m53s) | Lose(3m4s) | Win(3m53s) | Lose(2m42s) |

I intended this game to be challenging yet not too tiring time attack game which either can be updated into or a playing as a prequel of a bigger game. I added extra “Easter egg” scene which secretly implemented inside the game, and it’s loaded when a player defeats the game less than 3 minutes. The Easter egg scene will show new boss monster foreshadowing that there will be updates to the game. Two of my friends who already have many experiences in other similar type of games took about 4-6 trials to defeat the game for the first time, and one friend could not even defeat the game in 7 trials. Even those friends who cleared the game could not defeat the game in 3 minutes to see the Easter egg scene. I concluded that the level of game is hard enough for players to spend their some time to clear the game for the first time, and need to put harder efforts to master the game and clear the game in 3 minutes.

One of the friend suggest me to add a feature in the game to heal the player character so the player can have option to spend some time and heal the character or save the time and risk defeating the monster with lower hp. I took that as additional user requirement and added a “healing zone” in the middle of the map where if the player stays there, he can heals his hp. To make it more entertaining, I added a dog animation which stands still but if a player stays near him he bounces up and down.