GDD and TDD of the game 'Cast Away'

Name: Debasish Das

Student ID: 2006925

Game Design v1.1.0:

It is a single player Platform Game which can also be considered as a 'jump and run game'. The player is casted away in a 2D world where s/he needs to look for pearls and fruits. When the player finds all the pearls for a level, s/he goes to the wizard. The wizard makes a portal out of those tesseracts that takes the player to the next level #1.

There are spikes and moving traps. If the player comes to contact on those, or, if the player falls from too high (considering the falling velocity), s/he will lose, and has to start the same level again.

The moving traps move from left to right or, up to down #2. There can be gap from a piece of terrain to another.

The fruits help the player to get powerups #3. There will be one basic powerups which is 'Thrust Shoes'. 'Thrust Shoes' allows user to jump in the air #4. Power bar decides when the user can double jump. Power bar recharge depends how much fruit a player has #5.

Todo List:

1. Complete the yellow highlighted parts of the Game Design v1.0.1 marked from 1 to 5.
2. Design and Develop Level 2.
3. Remove unnecessary Terrain from Level 1.

Product Backlog or, Wishlist:

1. There will be moving platform.
2. The player might break different objects like rocks, boxes where pearls might be hidden.
3. Enemy will be there to track the player.
4. Player can destroy enemy using attack #1.
5. An infinite level might be introduced where the terrain to jump on, obstacles and fruits will be generated randomly.

Technical Design v1.1.0:

1. Player: The player is a rigid colliding object with gravity. It has the ability to move forward, backward and jump. The jump thrust has 20 unit, and horizontal movement has velocity of 9 unit. Moving backward flips the player's sprite. The Camera moves along with Player's movement. The jump is decided by checking if the player is on the ground by using 'Physics2D.BoxCast' function. The function uses the terrain layer with the box collider of the player facing downward. For double jump or jumping in the air, the jump count is used. Player animator triggers different animation for its movement. Player prefab also has audio source for jump, item collection, death and completing a level successfully.
2. Terrain: Terrain is a rigid colliding static body designed with Tilemap. The Tilemap uses 2D collider and spites sliced images. For performance efficiency, the terrain uses 'Composite Collider 2D'. It also uses 'Platform Effector 2D' so that user can not stick at the left and right side of the collider. The 'Use one way' option for 'Platform Effector 2D' is turned off so that user cannot break inside from downside of tiles.
3. Traps and Death: Traps are colliding body with tag 'Trap'. When player collides with traps the level restarts. Player's death sound and animation is triggered. Its body turns static. The restart level is triggered from death animation since its desirable to have some time after the player dies. Spike trap prefabs can also be faced downward and sideway. There are prefab of five spikes. These are all Variant Prefab. The moving trap transforms by x and y axis. It also has rotating behavior.
4. Item collection: Items are isTrigger colliding body. The number of items collected is updated and displayed in the canvas when a player collects them. If all the Pearl items are collected for a level, portal opens when the player collides with the wizard. Else, wizard pops up notification that all the pearls are not collected. Collecting fruits reduces the recharge time for double jump power bar. The double jump is set true after collecting 20 fruit points. Double jump is initially true. When player performs double jump it is set false. Then a function invoked to set it true. The timer depends on how many fruits has been collected.
5. Background: Background is created with tiles made from sliced sprites. Background music is different while a player is playing or s/he is start/end scene.
6. Start & End Scene: Start scene is used to start the game. End scene is used to exit the application or, starting the game again.

7. Enemy #1: Enemy will use path-finder algorithm to follow the player and animation for its movement. Enemy velocity will be slower than Player.