Amazon Outlivers

Prepared By:

A.Ranjith Kumar 15CS30002

A.Sai Krishna 15CS10003

K.S.Satish Kumar Reddy 15CS10018

Aravind Kollipara 15CS10020

C.Harsha Sai Reddy 15CS30011

Table of Contents

Objective:

The objective of this project is to develop a serious game which helps the Users to enhance strategic thinking, decision making skills through an imaginary survival scenario. It is designed to show that these skills can be trained, improved and developed in an effective and actual way.

Target Users:

This game is targeted for all people in the Age-Group 13 to 30 who have zeal to enhance and sharpen their innovative thinking and decision making skills.

Detailed Storyline highlighting each characters and their properties:

StoryLine:

The game starts with a plane crashing on an island. A person who survived the plane crash starts exploring the island for food, water,shelter. He should start building a boat using the resources available from the island. He should survive until the end of the game.He should kill the small animals in order to get food and he should escape the large animals He collect fruits,wood and store them for future use.

Characters:

The player controlled by the user is the the Main character. Other characters include Wild animals like Tiger, huntable animals like deer.

Specifications:

1. As he is staying in an island, they can face threat from animals. So they will be given weapons like axe,spear,bow and arrow according to their skills.
2. A health index will be shown on the right corner of the screen.
3. Apart from the health bar, there are 3 bars - stamina bar, thirst level bar and hunger level bar.
4. Health index will start decreasing at a constant rate and to maintain the health, the player needs to eat/drink in regular intervals.
5. Health index of a character decreases if any wild animal attacks.
6. Similarly health index increases if the character is able to find any food or kill any animal.
7. Excess food obtained will be stored for future purposes.
8. Smaller animals should be hunted for food while the character will hide or escape from the Larger animals.
9. Smaller animals try to escape from the player. So, the player should follow that animal until its health index drops down to zero(aka animal is dead).
10. If the player is near a wild animal by some minimum threshold, then the player's health would start decreasing at a fast rate. So, the player is supposed to run away as quickly as possible.
11. A player can chop palm trees only for collecting wood. By Chopping palm trees, he also gets coconuts.

Game Termination:

A Player will **lose** if his health drops down to zero.

To **Win** in the game, the player needs to collect all the items needed to build a boat. The faster he is able to collect all the items, the higher is his score in that game run.

Implementation:

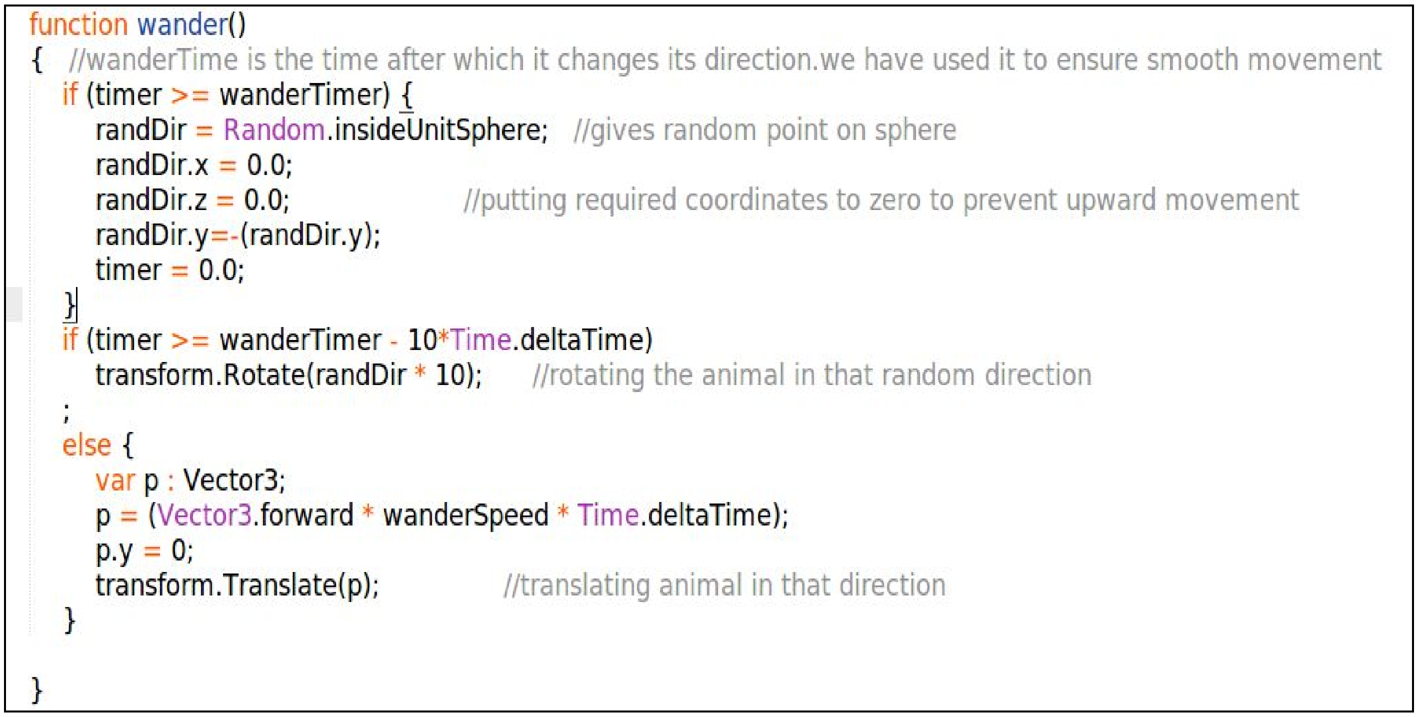
Design of AI Characters:

Animals:

The characters of this category serve as food for main characters or they will try kill the players.

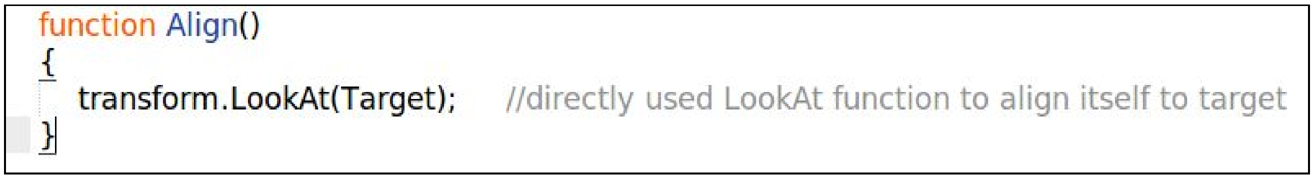
Algorithms Used:

- In case of wild animals, there are a couple of radius defined around it. One is the LookAt Radius and the other is the Attack Radius. When the player is out of LookAt Radius, the character exhibits **wander** behaviour.The code we have written looks like this



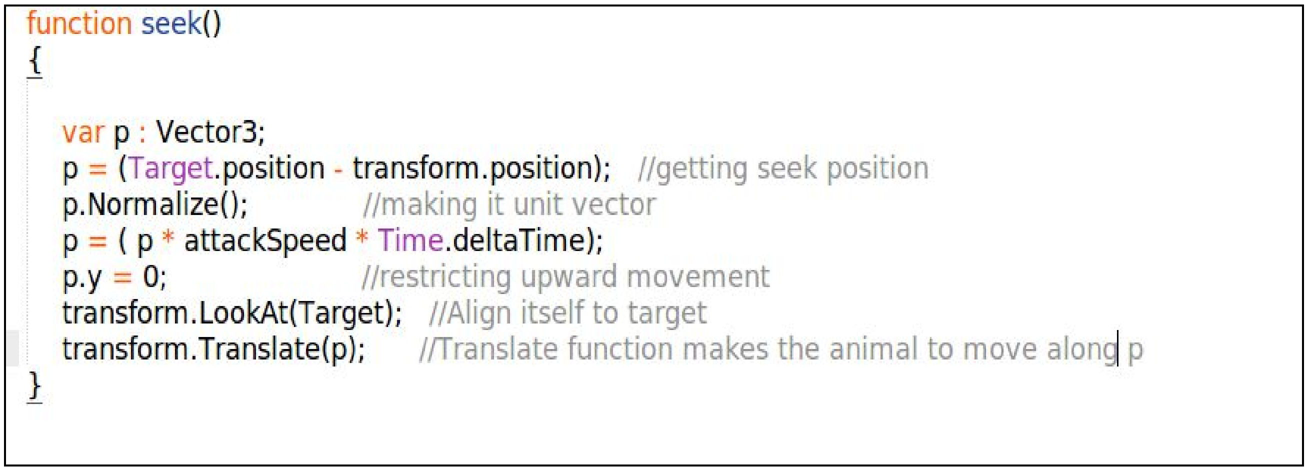
```
function wander()
{   //wanderTime is the time after which it changes its direction.we have used it to ensure smooth movement
    if (timer >= wanderTimer) {
        randDir = Random.insideUnitSphere;   //gives random point on sphere
        randDir.x = 0.0;
        randDir.z = 0.0;                //putting required coordinates to zero to prevent upward movement
        randDir.y=-(randDir.y);
        timer = 0.0;
    }
    if (timer >= wanderTimer - 10*Time.deltaTime)
        transform.Rotate(randDir * 10);      //rotating the animal in that random direction
    ;
    else {
        var p : Vector3;
        p = (Vector3.forward * wanderSpeed * Time.deltaTime);
        p.y = 0;
        transform.Translate(p);             //translating animal in that direction
    }

}
```

- When the players is between LookAt Radius and Attack Radius, then the animal **aligns** itself towards the player and roars towards it.



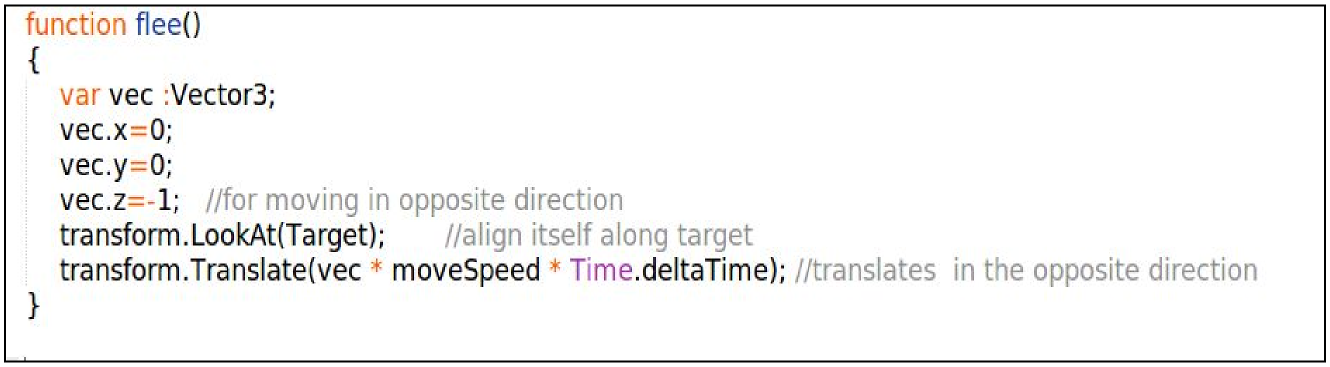
```
function Align()
{
    transform.LookAt(Target);     //directly used LookAt function to align itself to target
}
```

- When the player is under Attack Radius, then the animal **Seeks** to the player and attacks the player.



```
function seek()
{

    var p : Vector3;
    p = (Target.position - transform.position);   //getting seek position
    p.Normalize();            //making it unit vector
    p = ( p * attackSpeed * Time.deltaTime);
    p.y = 0;                  //restricting upward movement
    transform.LookAt(Target);   //Align itself to target
    transform.Translate(p);     //Translate function makes the animal to move along p
}
```

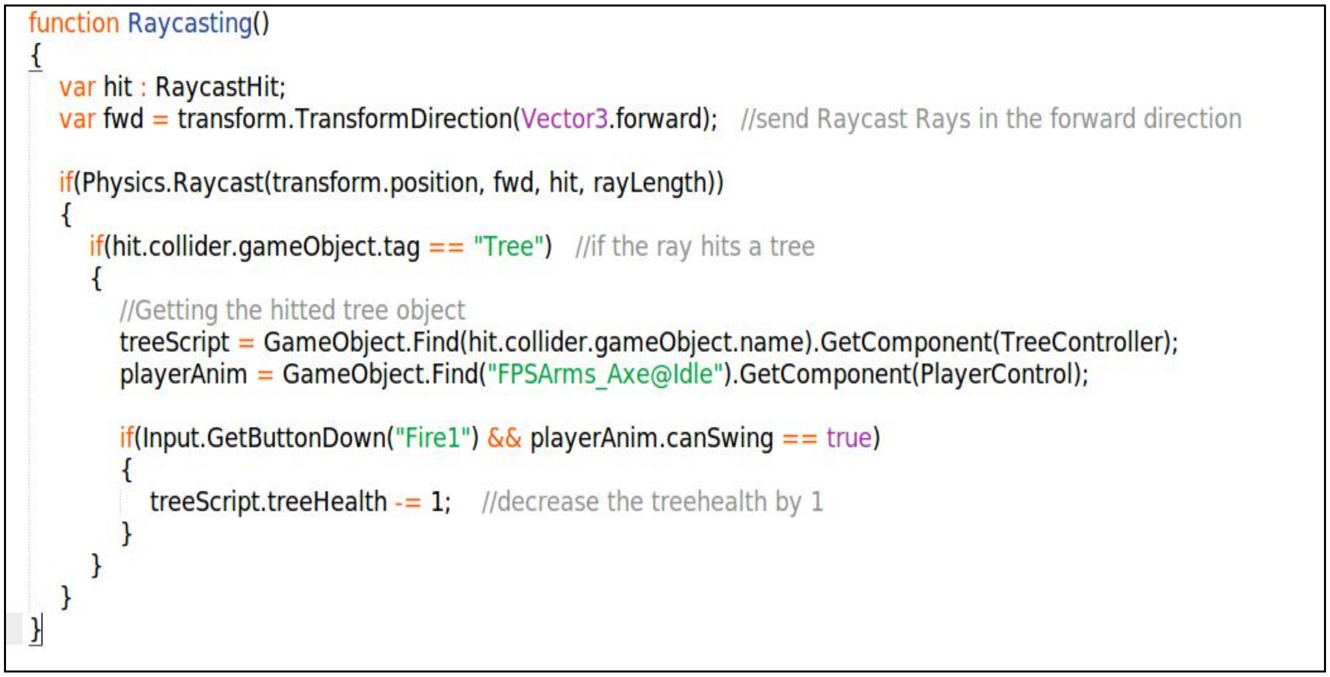
- When a player hunts the small animal,then the animal tries to escape from him/her using **flee** algorithm.



```
function flee()
{
    var vec :Vector3;
    vec.x=0;
    vec.y=0;
    vec.z=-1;   //for moving in opposite direction
    transform.LookAt(Target);       //align itself along target
    transform.Translate(vec * moveSpeed * Time.deltaTime); //translates  in the opposite direction
}
```

Tree Cutting:

- For detecting whether player is hitting the tree or not,we have used **Ray Casting** from Axe to detect the cutting.



```
function Raycasting()
{
    var hit : RaycastHit;
    var fwd = transform.TransformDirection(Vector3.forward);   //send Raycast Rays in the forward direction

    if(Physics.Raycast(transform.position, fwd, hit, rayLength))
    {
        if(hit.collider.gameObject.tag == "Tree")   //if the ray hits a tree
        {
            //Getting the hitted tree object
            treeScript = GameObject.Find(hit.collider.gameObject.name).GetComponent(TreeController);
            playerAnim = GameObject.Find("FPSArms_Axe@Idle").GetComponent(PlayerControl);

            if(Input.GetButtonDown("Fire1") && playerAnim.canSwing == true)
            {
                treeScript.treeHealth -= 1;    //decrease the treehealth by 1
            }
        }
    }
}
```

- Also used raycast for implementing pickup option after chopping tree into logs and coconut.

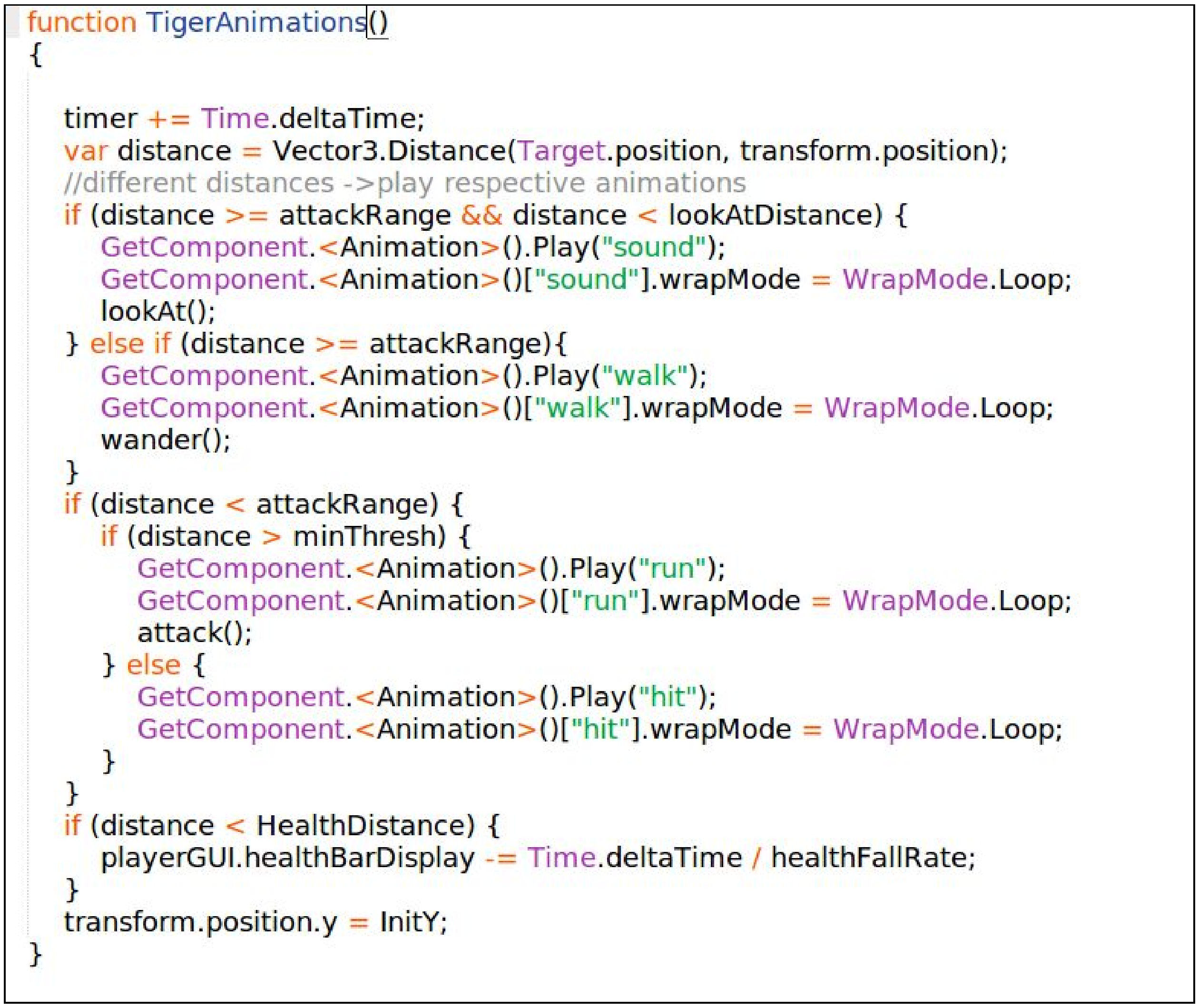
Inventory:

- There will be a central inventory in which the user collected items will be stored.The Player can use the inventory items to increase his health,stamina and decrease his hunger and thirst.

Animation Codes used in the Game:

Tiger:

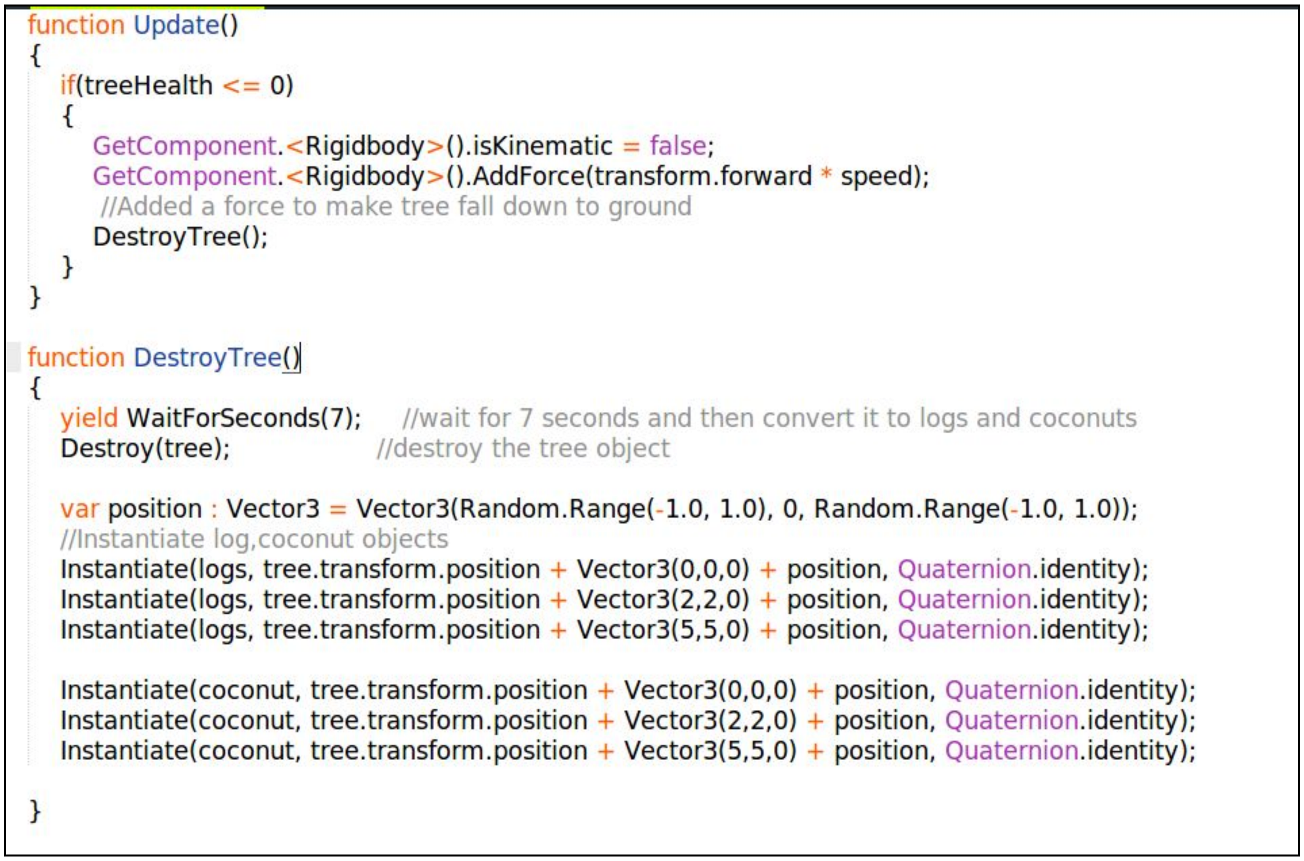
- For wild animals like tiger we have used animations like Idle,Walk,Run,Roar,Hit when it is in different radius ranges(as said above).



```
function TigerAnimations()
{

    timer += Time.deltaTime;
    var distance = Vector3.Distance(Target.position, transform.position);
    //different distances ->play respective animations
    if (distance >= attackRange && distance < lookAtDistance) {
        GetComponent.<Animation>().Play("sound");
        GetComponent.<Animation>()["sound"].wrapMode = WrapMode.Loop;
        lookAt();
    } else if (distance >= attackRange){
        GetComponent.<Animation>().Play("walk");
        GetComponent.<Animation>()["walk"].wrapMode = WrapMode.Loop;
        wander();
    }
    if (distance < attackRange) {
        if (distance > minThresh) {
            GetComponent.<Animation>().Play("run");
            GetComponent.<Animation>()["run"].wrapMode = WrapMode.Loop;
            attack();
        } else {
            GetComponent.<Animation>().Play("hit");
            GetComponent.<Animation>()["hit"].wrapMode = WrapMode.Loop;
        }
    }
    if (distance < HealthDistance) {
        playerGUI.healthBarDisplay -= Time.deltaTime / healthFallRate;
    }
    transform.position.y = InitY;
}
```

Tree:

- After cutting a tree,the tree will fall to the ground and some logs and coconuts will appear on the ground.For doing this the script is as follows



```
function Update()
{
    if(treeHealth <= 0)
    {
        GetComponent.<Rigidbody>().isKinematic = false;
        GetComponent.<Rigidbody>().AddForce(transform.forward * speed);
         //Added a force to make tree fall down to ground
        DestroyTree();
    }
}

function DestroyTree()
{
    yield WaitForSeconds(7);     //wait for 7 seconds and then convert it to logs and coconuts
    Destroy(tree);               //destroy the tree object

    var position : Vector3 = Vector3(Random.Range(-1.0, 1.0), 0, Random.Range(-1.0, 1.0));
    //Instantiate log,coconut objects
    Instantiate(logs, tree.transform.position + Vector3(0,0,0) + position, Quaternion.identity);
    Instantiate(logs, tree.transform.position + Vector3(2,2,0) + position, Quaternion.identity);
    Instantiate(logs, tree.transform.position + Vector3(5,5,0) + position, Quaternion.identity);

    Instantiate(coconut, tree.transform.position + Vector3(0,0,0) + position, Quaternion.identity);
    Instantiate(coconut, tree.transform.position + Vector3(2,2,0) + position, Quaternion.identity);
    Instantiate(coconut, tree.transform.position + Vector3(5,5,0) + position, Quaternion.identity);

}
```

- Some other minor script includes Player hands animation,inventory script etc

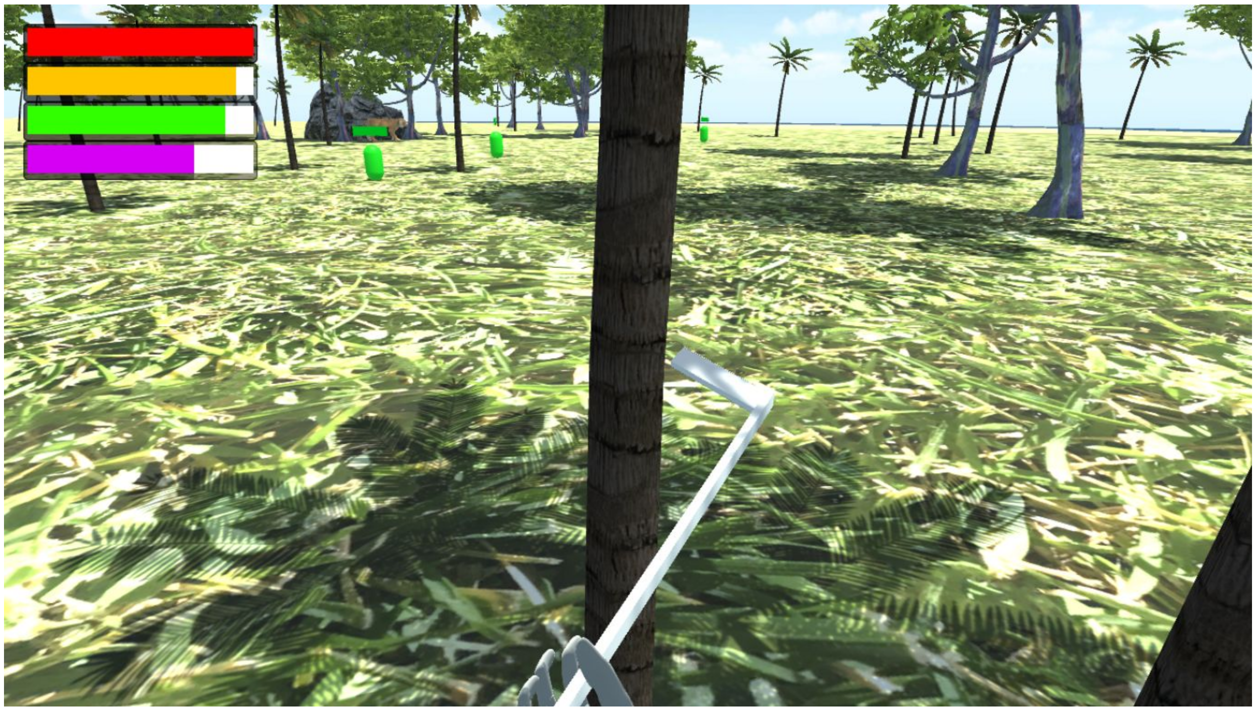
Scenes in the Game

Major scenes in the game includes the following

- Wood Cutting
- Escape from the wild Animals
- Wander Scenes
- Picking up logs

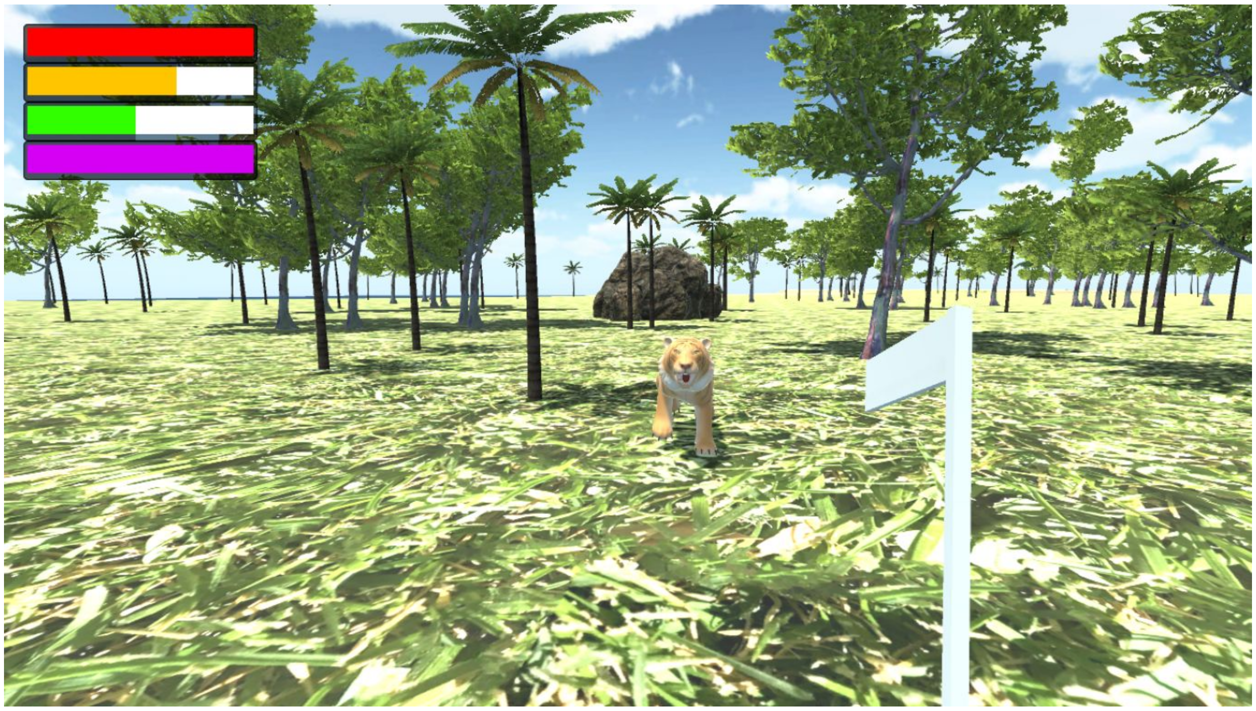
Wood Cutting:

Trees will be placed randomly on some locations of the map.Whenever the player sees a tree on the map and wants to cut it for wood,he just need to click on the tree and if the inventory contains Axe,the character starts cutting the tree and the wood will automatically placed in the inventory.

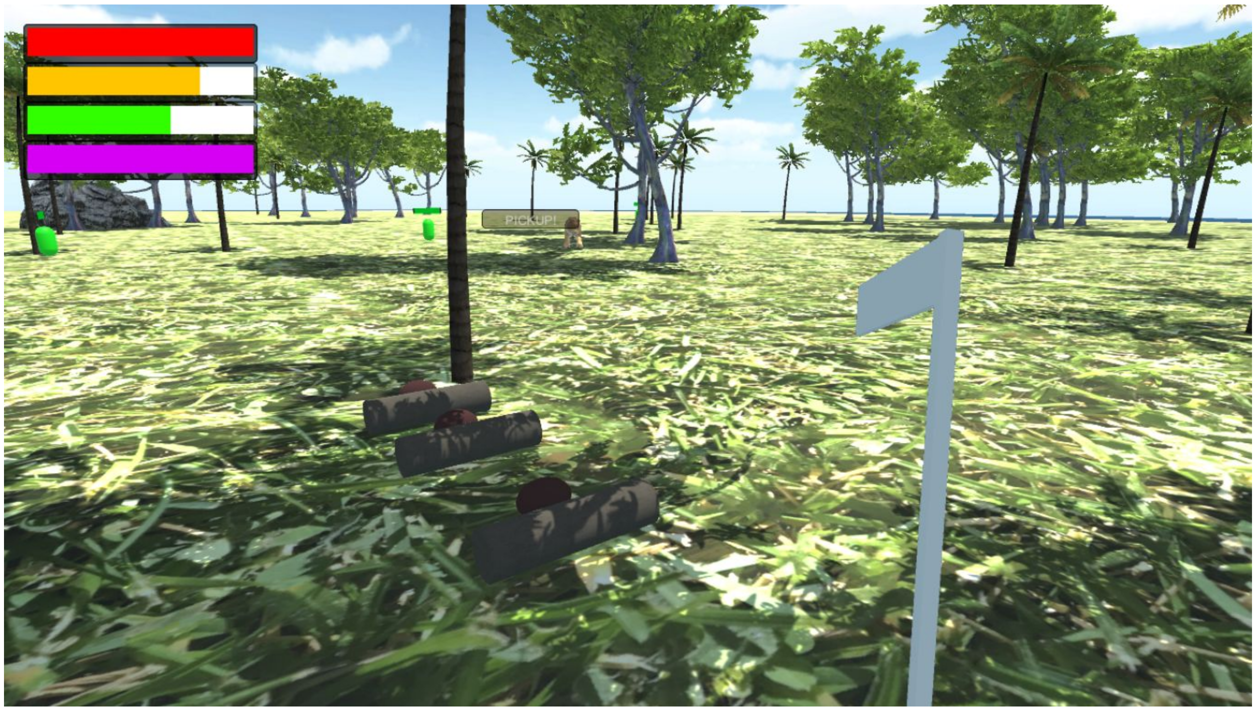


Escape from wild Animals:

Whenever a character goes into sight of wild animal,it will try to kill the character,the character should run away in order to save his health.The scene will look something like this



Picking up logs:



Wander Scenes: