**SURVIVE ON ISLAND**

# Introduction:

Survive on Island is a third-person survival game that takes place on a highly secured island and the player is trapped in it. The player will have to find weapons, food and kill guards to survive on the island for the longest time possible.

# Game Mechanics:

## Player Movement:

The player will be able to walk or run around the environment and jump on small platforms to reach higher areas.

## Combat:

The player will be able to use melee weapons found in the island to fight against guards patrolling the area.

## Enemy AI:

The enemies will patrol specific areas and engage on combat once the player is in their sights.

## Animal AI:

The animals simply wander around the island and flee when attacked by the player.

## Survivability:

The player will have to hunt for food or kill enemies and loot food from the corpses to stay alive.

## Score Based on Time:

The final score is determined on how long the player has survived in the island.

## Camera Movement:

The camera will follow the player in third person and can be rotated in yaw and pitch directions.

## Lose Condition:

The player will lose the game once it loses all its health.

# Assets:

## Static Objects:

1. **Trees:** The stem of the tree will be in brown cylindrical shape and the leaves will be in a spherical shape placed on top of the stem. The player’s movement is restricted when it collides with a tree. The player can also cut down a tree to obtain a weapon.
2. **Bushes:** These will be scattered around the island as green cubes. The player will be able to walk through it and crouch to hide from enemies’ line of sight.
3. **Boulders:** These are also scattered around the island as black cubes. They simply exist like walls and will restrict player/entity’s movement axes when collided with.

## Dynamic Objects:

1. **Movable Boulders:** These can be distinguished from stationary boulders by their lighter black colour. They are also in the shape of cubes and can be moved by the player. The boulder’s position will mimic the player’s position when moved.
2. **Pebbles:** These are represented by tiny black spheres that roll over the ground when the player walks on it. A small force is applied to these pebbles upon collision that allows them to roll in the direction the player is moving.

## Collectibles:

1. **Weapons:** The weapons are identified as long brown colour cuboids which are also scattered around the island. These can be used to attack soldiers or hunt animals. Once the player has picked up the weapon, the weapon’s position will be set to the player’s hand position and will update the HUD with the durability of the equipped weapon.
2. **Food:** Foods can be found in various shapes and colours representing different kinds of fruit and having different healing attributes. Foods are automatically picked up and stored in the player’s inventory which can be seen in the HUD. The inventory will be an array of 3 food assets and the player has a choice to eat whichever food in the inventory. I will display what some of the fruits do in the flowchart below.

## Particles:

1. **Leaves:** Leaves are sometimes found to be falling from trees or blown by the wind. They’re going to be represented as flat thin cubes. A particle class will be created to load a single leaf asset and add some texture to it. Another class will be created to spawn multiple instances of the leaf created by the previous class and have different methods for random movement.

# Camera:

Since this game is played in third person, the whole body of the player will be seen and will follow the player within 5 meters or so.

The camera’s position is updated along with the player’s position and this will ensure that the camera is following the player always. This is performed by calling the XMMatrixLookAtLH function from the XNA Maths library which takes the camera’s position, target position and the direction that faces up as parameters.

The camera can also be rotated around the player in pitch and yaw directions. This action is executed by calling the XMMatrixRotationRollPitchYaw function from the same library that takes the rotation values for all 3 axes as parameters. The user can move the camera using the mouse.

# User Interface (UI):

The main menu is going to consist of a play button, an options button, a leaderboards button, and an exit button. The play button will allow the user to enter the game and start playing. The options button will allow the user to adjust technical features such as sound, shadows, field of view (FOV), etc. The leaderboards button will allow the user to view the scores earned by players and for how long they have survived. The exit button will simply exit the game.

A click on the button will call a function the button is referring to and the function will execute the appropriate action(s). The options menu is going to have different pages for different settings and all these pages will be categorised as functions which are responsible for modifying the settings. During the process of transitioning from one page to another, a separate function will be called to stop displaying the current page and display the new page.

The in-game heads up display (HUD) will contain the player’s health, its energy, the equipped weapon’s durability, the number of fruits carried, the score earned and the time elapsed. Each UI element will receive their respective info from various classes and display them spontaneously.

All the text in the game is going to be rendered by using each letter as assets.

# Lighting:

The game is going to consist of a single light source which is the sunlight. It is going to illuminate almost the whole part of the island with a bright yellowish-orange colour. The process of simulating light is a slightly complex process but to describe it in simple terms, light consists of 3 terms: ambient, diffuse, and specular. We’ll being using all these terms to produce light in our game and I will now explain how each term is going to be calculated.

The ambient term consists of just a single colour which can be added to the final colour. The diffuse term is determined by calculating the dot product of the light vector and the normal of the surface. The specular term is calculated by the value of the dot product raised to the power of a certain integer. The final colour of the light is established by multiplying all the 3 terms.