NP COMPLETE

Tree Huggers

Design Document

Zach Ison
Kevin Miller
Darren Ng
Doug Smith
Matt Tatoczenko
Marshall Williams

Table of Contents

Introduction

Tree Huggers is a semi-realistic exploration and survival game that is set in a forest environment. The initial idea for the game came about due to a desire to create a very realistic and physically correct game, almost like a wilderness simulation. By being set in the forest, a lot of stunning visuals could be implemented to make a player feel like they were out in the forest themselves.

Tree Huggers will be a single player first-person game. It incorporates a full day and night cycle with a realistic looking sun and moon. 3D sounds will be used to add to the immersive element of the game. An inventory system will be used to hold various materials and weapons that the player will use. There will be AI animals, both as predators and neutral agents, to allow some additional interaction for the player. Finally, there are player attributes that include health, stamina, hunger, and thirst, all of which will affect player motion and could possibly lead to the player's death.

The goal of Tree Huggers is to survive as long as possible while taking in the beautiful environment. The player will explore the beautiful landscape, fight off predators, and hunt other animals in order to survive the expedition.

Overview

Visual Style

Tree Huggers will try to be as visually correct as possible. High resolution art assets have been purchased to make the game look as great as possible. These assets include trees, rocks, other plants, and ground textures. Since the game is trying to be set in a realistic environment, there will be a physically correct sky that includes a sun, a moon, clouds, and weather effects. The game will have a day and night cycle with correct lighting color and intensity to make the player feel like they are out in a forest.

Controls

Tree Huggers will utilize Unity's first-person controller asset for the first-person camera. All of the actions will be relative to this first-person viewpoint. Crouching, sprinting, and combat have been added to the first-person controller to aid in player survival.

Player Attributes

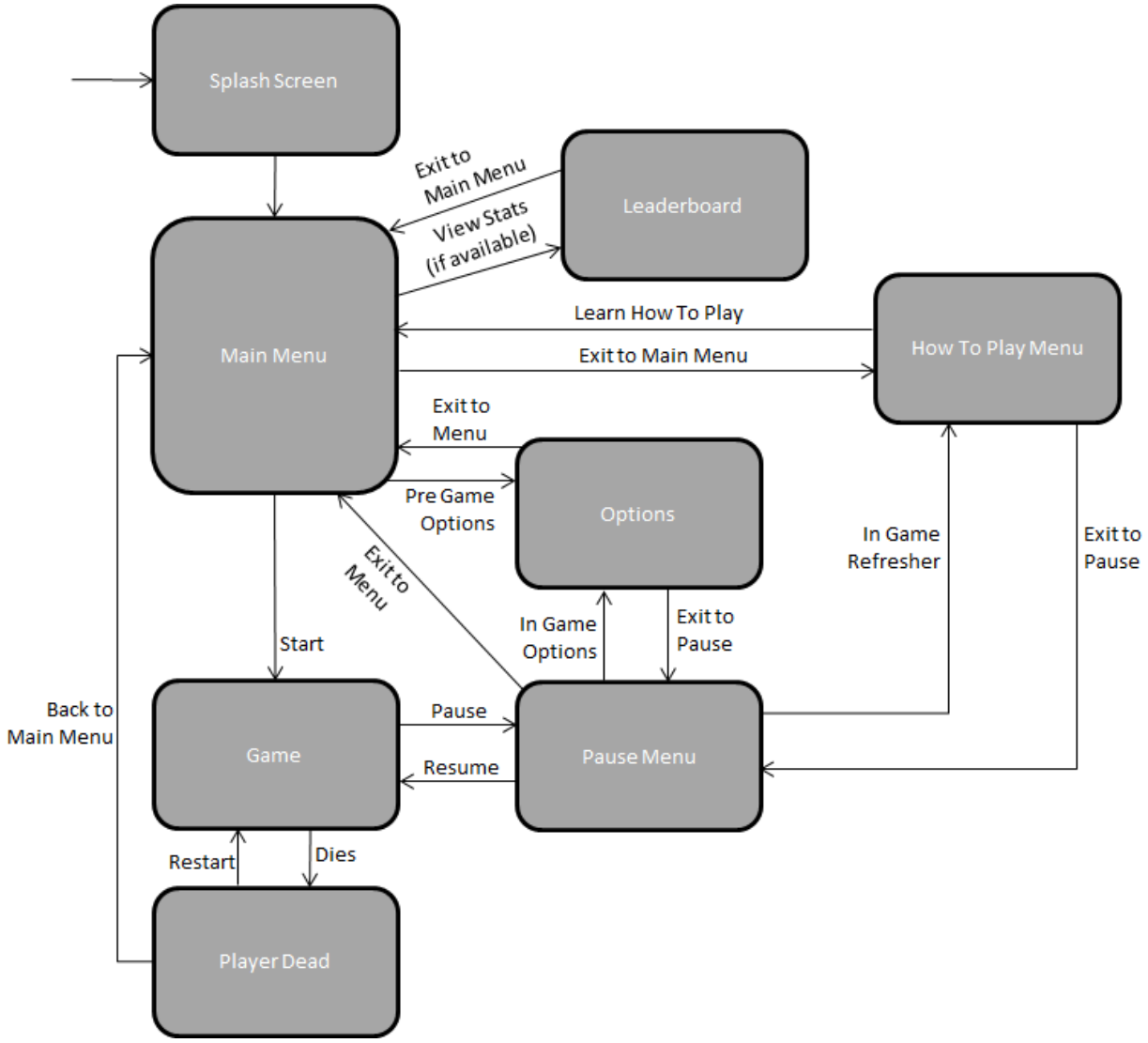
Instead of just having a health based system, Tree Huggers will incorporate other aspects that the player must be aware of to survive. Main health is a factor that will determine whether the player is still alive or dead. A stamina system will be used to limit running and jumping for the player. A hunger and thirst system will also be included that will affect both the player health and stamina.

Hunger and thirst are both tied into the day and night cycle to add realism to the game. The thirst meter will deplete completely after two days without water, after which the health bar will deplete. The health bar, if no health packs are used and no water is drunk by the player, will deplete one day after the thirst meter is fully depleted. This gives the player three full days to survive without water if no other damage is done to the player. The hunger meter will deplete after four days without meat and will deplete the health bar after that point as well. With the same effect as the depleted thirst meter, a depleted hunger meter will fully deplete the health bar after two days. The player can survive six days without food if no other damage is done to the player.

Animals

Tree Huggers will try to include as many forest animals as possible. These animals include birds, rabbits, bears, stags, boars, foxes, and wolves. All of the animals are purchased assets that the NP-Complete team has attached scripts to in order to move the animals and make the animals look lifelike. The wolf is the only predatory animal in that it will hunt the player during the night. The other animals will remain neutral to the player as long as the player does not get close to them. Some of the animals, like the bear and boar, will be more territorial, while the stag will have some random points of aggression towards the player. The rest of the animals will flee from the player if the player gets too close to them.

Game States


Splash Screen
Exit to Main Menu
View Stats (if available)
Leaderboard
Main Menu
Learn How To Play
Exit to Main Menu
How To Play Menu
Exit to Menu
Pre Game Options
Options
In Game Refresher
Exit to Pause
Exit to Menu
Start
In Game Options
Exit to Pause
Back to Main Menu
Pause
Game
Resume
Pause Menu
Restart
Dies
Player Dead

Player

Controls (Keybindings)

While most of the controls can be rebound in the options menu, here are the initial buttons used in Tree Huggers. The only controls that cannot be rebound are WASD for movement and the Space Bar for jumping, as those are tied to scripts from the Unity Standard Assets package containing the first-person character controller used.

W	Forward movement
A	Left movement
S	Backward movement
D	Right movement
E	Interact or pick up objects
I	Enter/Exit crafting menu
M	Enter/Exit map menu
X	Set waypoint marker
Space Bar	Jump
Left Shift (Hold)	Sprint forward
Left Control (Hold)	Crouch
Mouse	Look orientation
Left Mouse (in game)	Attack with weapon
Left Mouse (in menu)	Select item/menu option
Middle Mouse (in inventory)	Drop item
Mouse buttons (in inventory)	Drag item in inventory
Escape	Enter/Exit pause menu

Options

To allow the player to customize their game a bit more, custom keybinding was introduced in the options menu. In it, the player can change all of the keys except for the keys for general movement, as well as the key for jumping. There is also the option to return all of the keys to their default bindings. Here's an image of the control options menu.

The binding script is primarily a script in between the Unity input manager and the scripts that have been written for Tree Huggers. By having this in between system, it is easy to reassign keys and save the new keys in the keybinding class, that way no changes need to be done in other scripts in order to change key presses. Another great feature is that keybinding changes will work in any build of the game, allowing even web build players to change their keys.

HUD

The player HUD is kept very simple, that way it does not detract from the gameplay. Here's a sample image of the HUD in the game.

The slider bars in the bottom left represent the player stats. The top slider, which is red, represents the player's health. The next slider, which is green, represents the player's stamina, which depletes when the player sprints or performs evasive maneuvers. The following slider, which is yellow, represents the player's hunger and depletes over time. The final slider, which is blue, represents the player's thirst and also depletes over time.

The slider bars in the bottom left will fade away if no actions deplete the health or stamina bar. If inventory items are used to replenish any of the sliders, the sliders will appear on the screen to indicate that they have been updated. Having the sliders fade away gives the player more screen space to immerse themselves into the game. The sliders will come back onto the screen when the player is in the inventory screen.

The reticle in the middle of the screen is used as an interaction point between the player and objects in the environment. When the player is looking at a weapon or at a regular inventory item and is close enough to pick it up, the reticle will change to yellow. If the player is looking at an animal enemy, the reticle changes to red.

The top right of the screen has a compass, which is used to provide direction to the player based on which way they are looking. The compass also has a small red mark attached to it, which is a waypoint marker. This marker can be used to mark a key location, that way the player can find their way back to that location if they venture away from it.

The bottom right of the screen holds a top-down view of the player and their surrounding area. This is mainly useful for when the player is in combat with the animals of the forest, that way the player can see how far away the animals are from them.

Combat

There are two main weapon types in the game: a ranged weapon and a melee weapon. A hunter or explorer travelling to a remote area would have at least one of these, so one of each weapon type can be used for combat. The weapons are used to fight off the animals in the game as well as to hunt, as the only way to get food is to hunt.

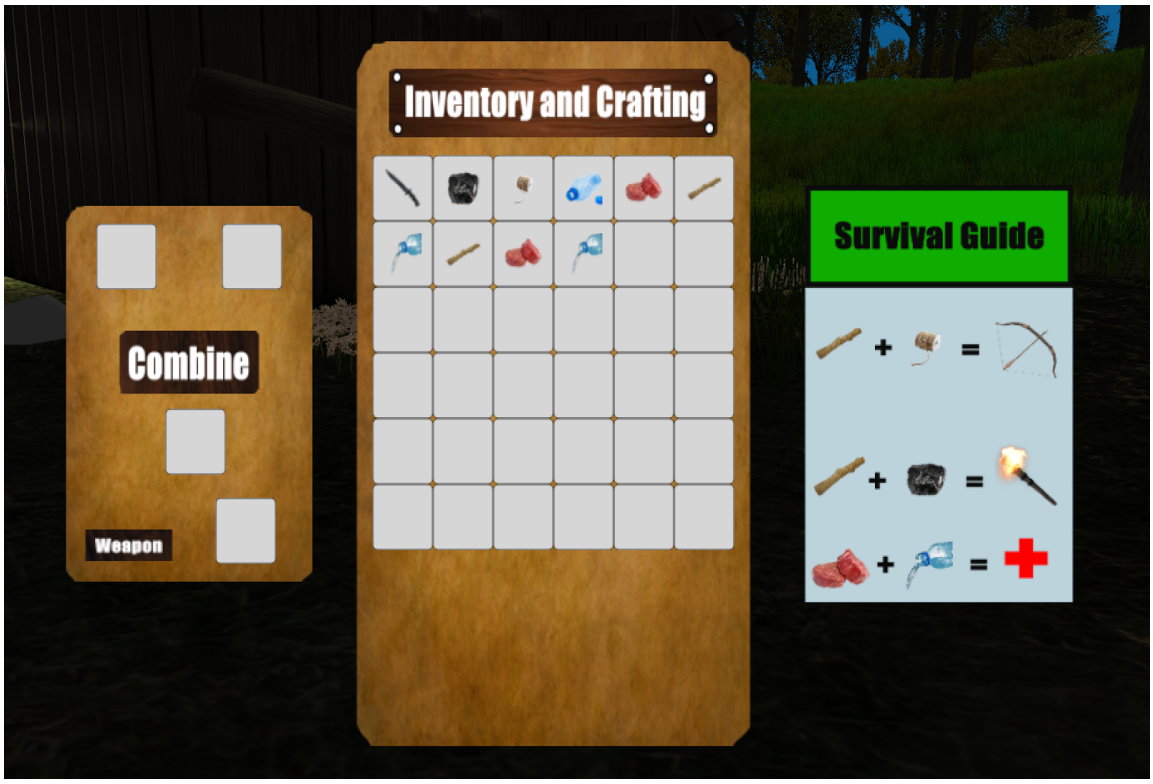
The melee weapon is a simple combat or hunting knife and is held on the right side of the screen for the player. The knife does a horizontal slash from the right side of the screen to the left side of the screen when the player decides to use it. The knife was the first weapon in the game, but it is hard to use as it has a short range.

The ranged weapon is a bow that fires arrows. The bow has an unlimited number of arrows that it can fire, making combat easier for the player. The bow must be charged for a short amount of time before it can fire an arrow, but the charge time is only one second, so arrows can still be fired quickly.

The bow has the added feature of including aim assist when firing arrows at the animals. Without aim assist, arrow firing needed to be precise, which is hard to do when facing moving targets. The aim assist feature does not move the player's reticle towards a target, however the looking speed of the character is reduced when they are looking at an animal. When an arrow is fired and the player's reticle is red, meaning the player is looking directly at an animal, the arrows will have a trajectory that takes them to the animal. In this way, if you are aimed at an animal and the reticle is red, the arrows will always hit the animal. This will help out novice players as well as players facing a large amount of enemies at once, as missing targets can prove fatal to the player.

Inventory/Crafting

In order to pick up, hold, and use items in Tree Huggers, a crafting and inventory system was introduced. Here is a screenshot of the menu system.

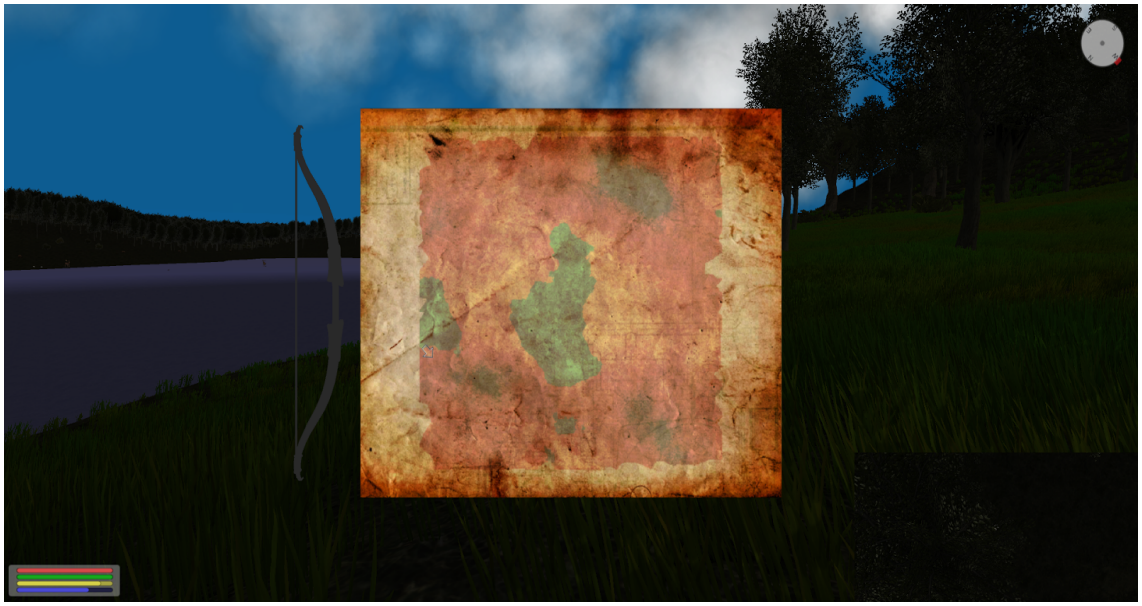


The menu uses a drag-and-drop style to place the items into the crafting area, located on the left side. Preferably, the right mouse button should be used to drag-and-drop, as the left mouse button uses items and the middle mouse button drops items. Items can be combined based on the small survival guide on the right side of the menu, where the items needing to be combined are put in the top two blank squares in the crafting section, then the output of them is gathered from the empty square below the “Combine” button.

To equip a weapon, drag a weapon to the empty slot to the right of the “Weapon” title. If a weapon is already equipped, that weapon must first be unequipped before a new weapon can be equipped.

Map

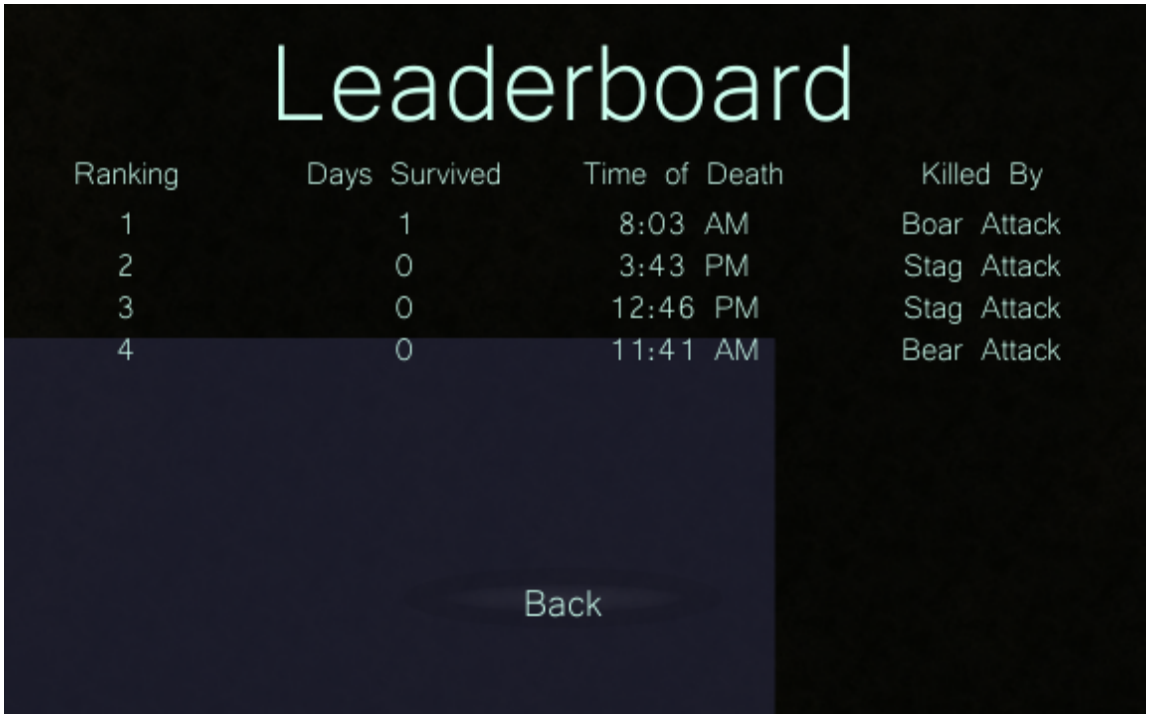
Since Tree Huggers is meant to be played from the perspective of an explorer in a forest setting, it makes sense that the explorer would have a map of the area. Take a look at the map from the game.



The map is made to look like it is made on a parchment style paper. It provides a basic view of the landscape, which is in the dull red color, and the water, which is in the blue-green color. By showing the general shape of the landscape, the player can determine their general location without having to see all of the terrain details, like trees and other shrubs. There is also a small arrow on the map which is used to show the player's position.

Leaderboard

In order to add a survival aspect, a leaderboard system was implemented. The leaderboard tracks the number of days survived, the time of day the player was killed, and what killed the player. A local copy of the game will save these stats locally, that way the player can check their playthroughs of the game. Here's an image of the leaderboard.



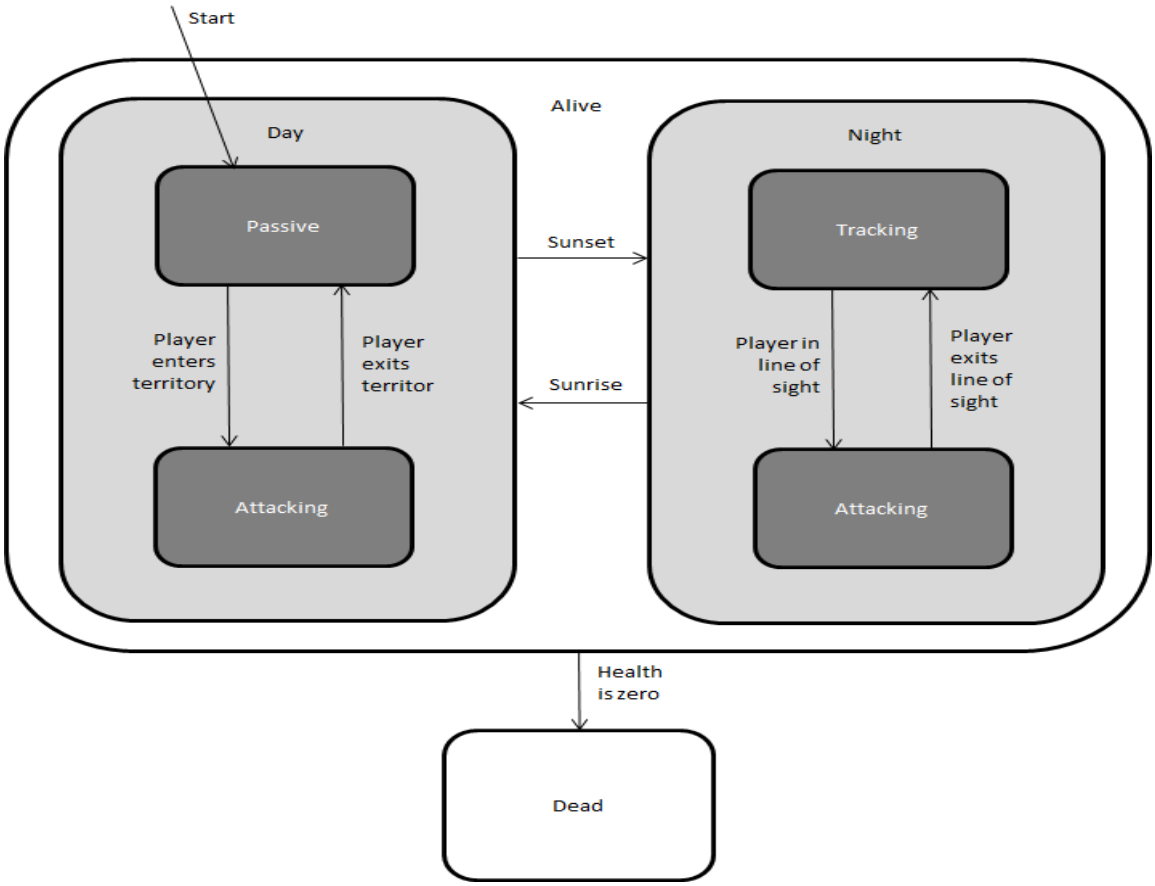
The leaderboard will only appear if the game has already been played and is accessed only from the main menu. It will show up to eight playthroughs, sorted based on days survived, then by time of death.

AI

Enemies

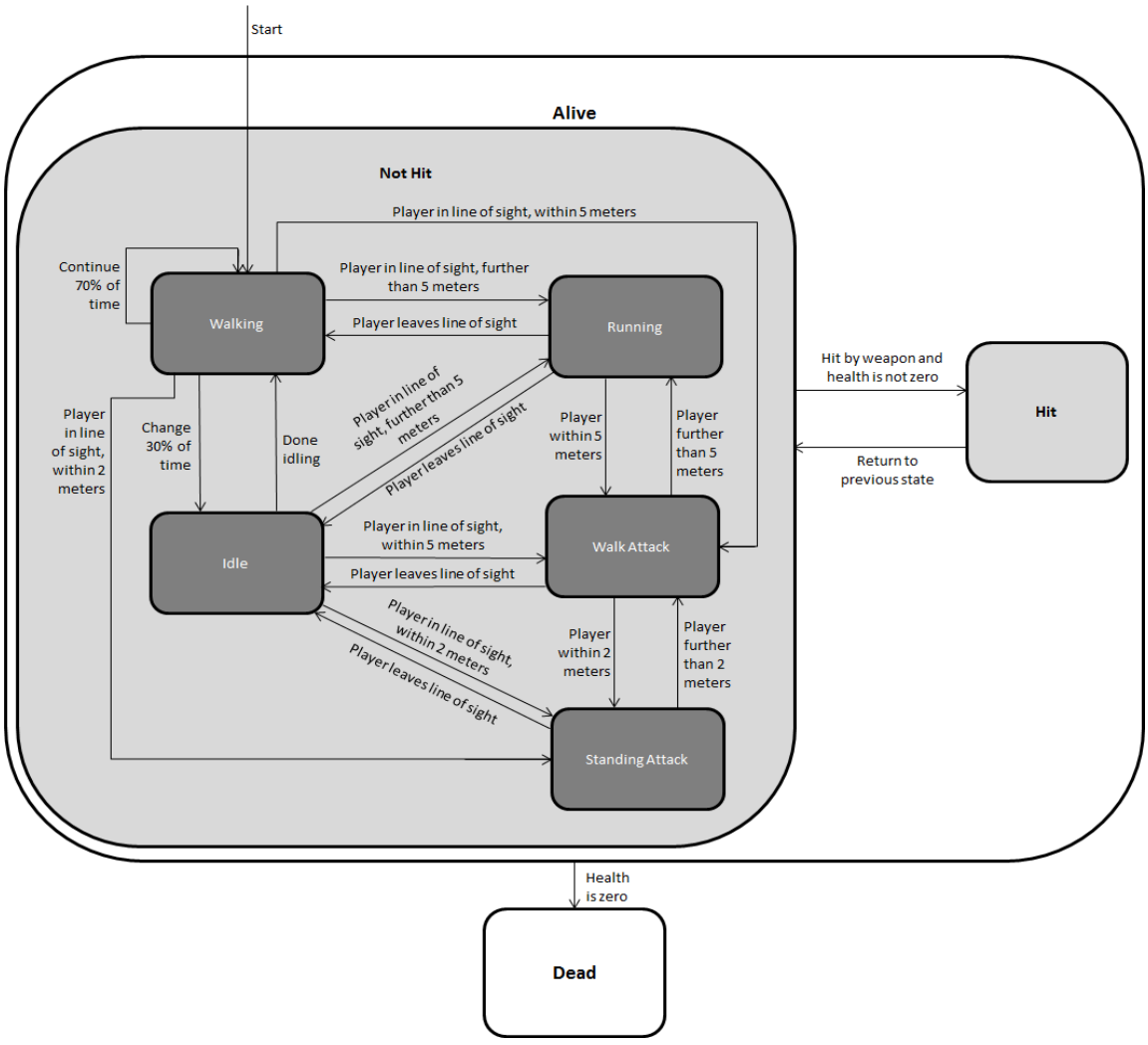
Wolf

The main enemy in the game is the wolf. The wolf utilizes simple states that change whether it is day or night. During the day, the wolf will attack the player if the player gets within a certain distance of the wolf. At night, the wolf will hunt down the player no matter where the player is at. Below is the state machine for the wolf.



Boar

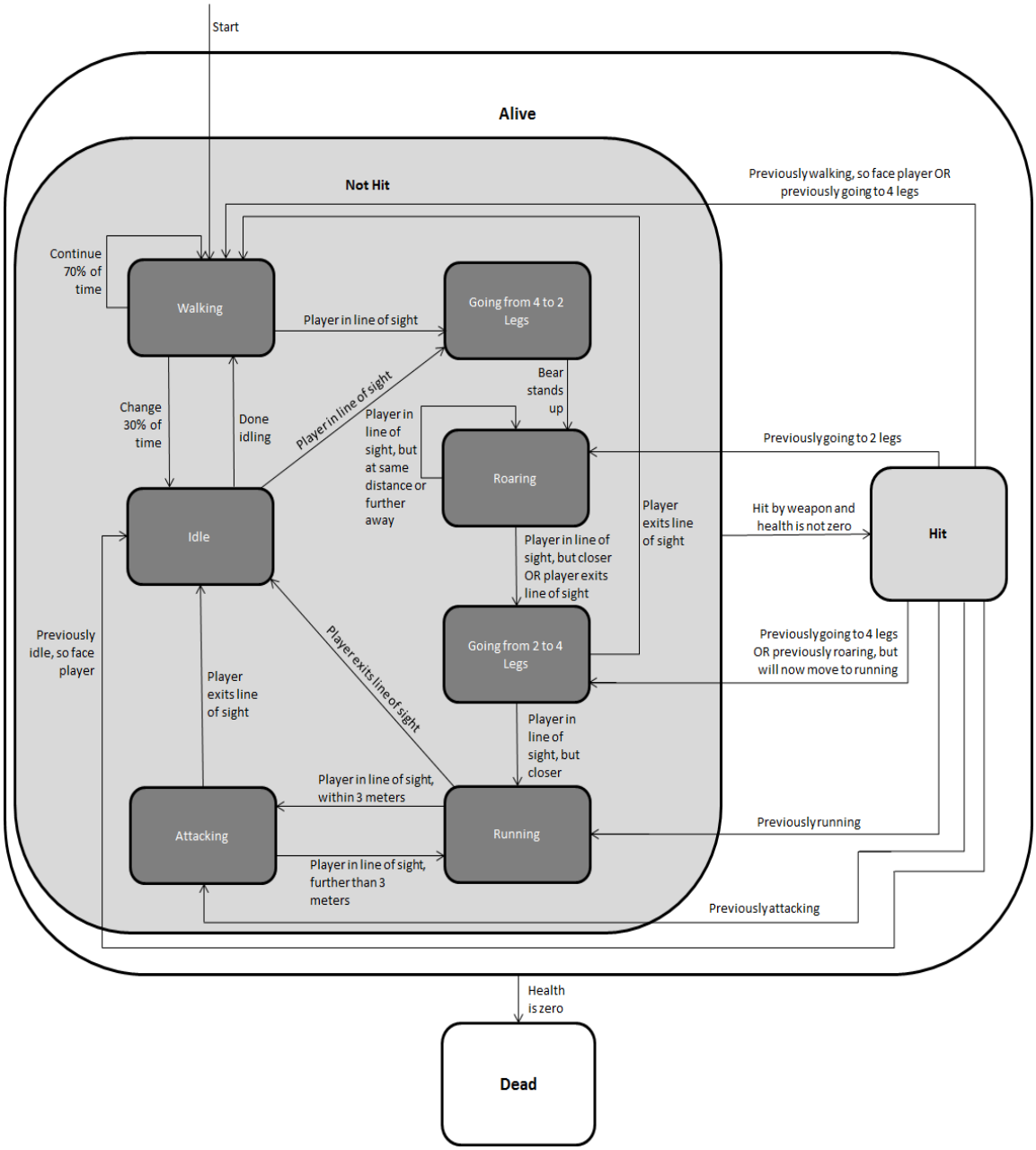
Another animal that is completely aggressive to the player is the boar. The boar uses state logic in its actions and mainly uses a line of sight and vision cone to detect if it can see the player. If the boar sees the player, it will run at the player and attack the player. If the boar is hit by an attack from the player, it will turn to face the direction that the hit came from before returning to the state that it was previously in. Below is the state machine for the boar.



Bear

The next animal that can be considered an enemy is the bear. The bear also uses state logic, but has a few modifications that make it different from the boar. The bear will roar at the player if it sees the player, but will only attack the player if the player moves closer to it after it roars. If the player backs out of the line of sight of the bear, the bear will return to a passive state again. The bear mostly wants to scare the player away, but will attack if provoked.

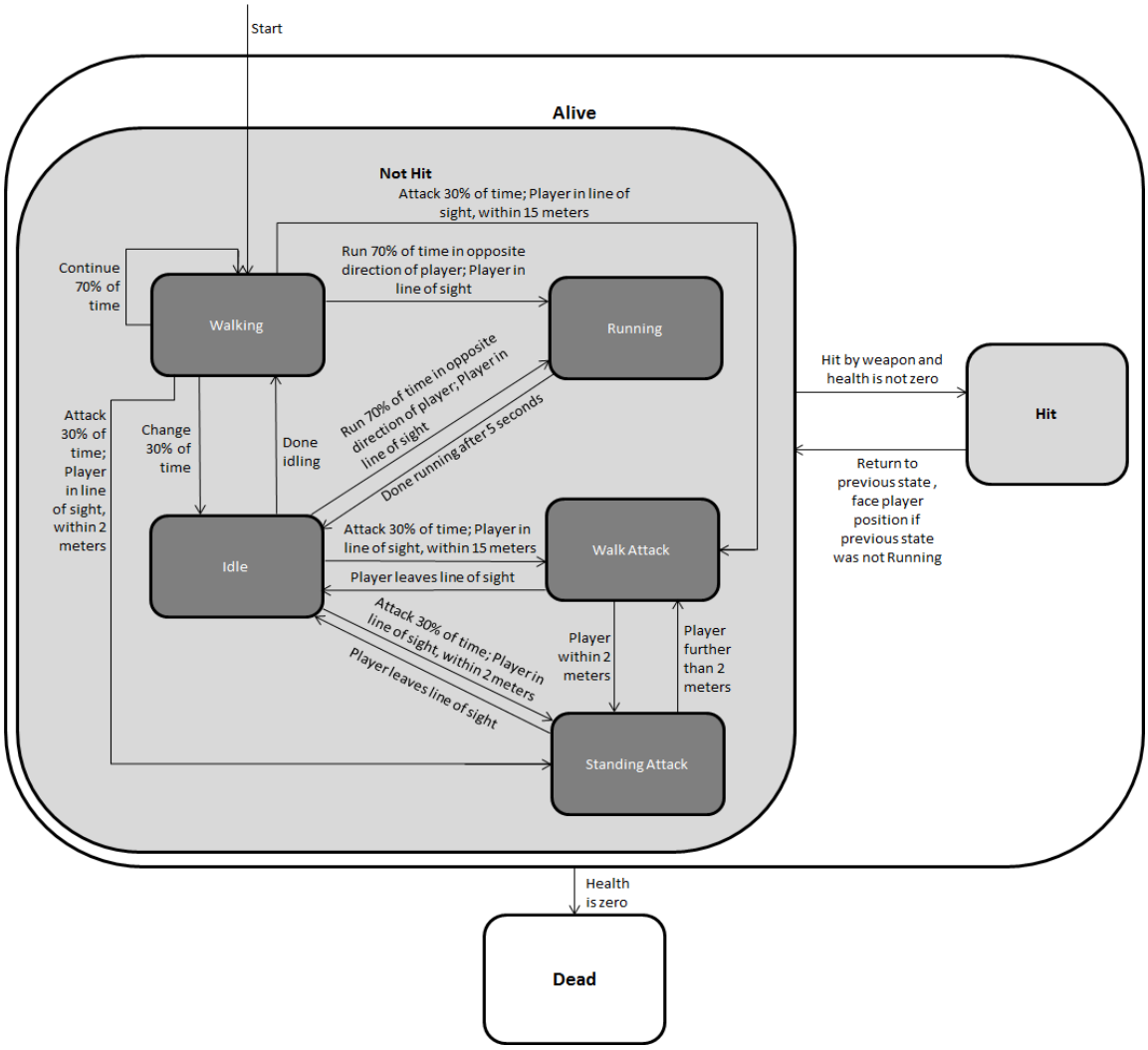
The bear also differs in the way it responds to being hit by the player. If the bear has not seen the player but is hit, it will turn to face the player. If it then sees the player, it will roar at them. If the bear is roaring at the player and is hit, it will move to attack the player. If the bear is already attacking the player, it will keep attacking the player.
To get a more detailed look at the bear, see the state machine on the following page.



The bear was the most complicated animal, as it was not made to just attack the player on sight. By having the roaring ability as well as moving from two legs to four legs and back again, the bear become complicated. As such, its responses to being hit could not be the same as the other animals. Instead of returning to previous states, the bear tries to move to an aggressive animal when hit by either facing the player or by moving to the next state in the state machine. Overall, this complicated state machine and logic helped to make the bear seem as realistic as possible.

Stag

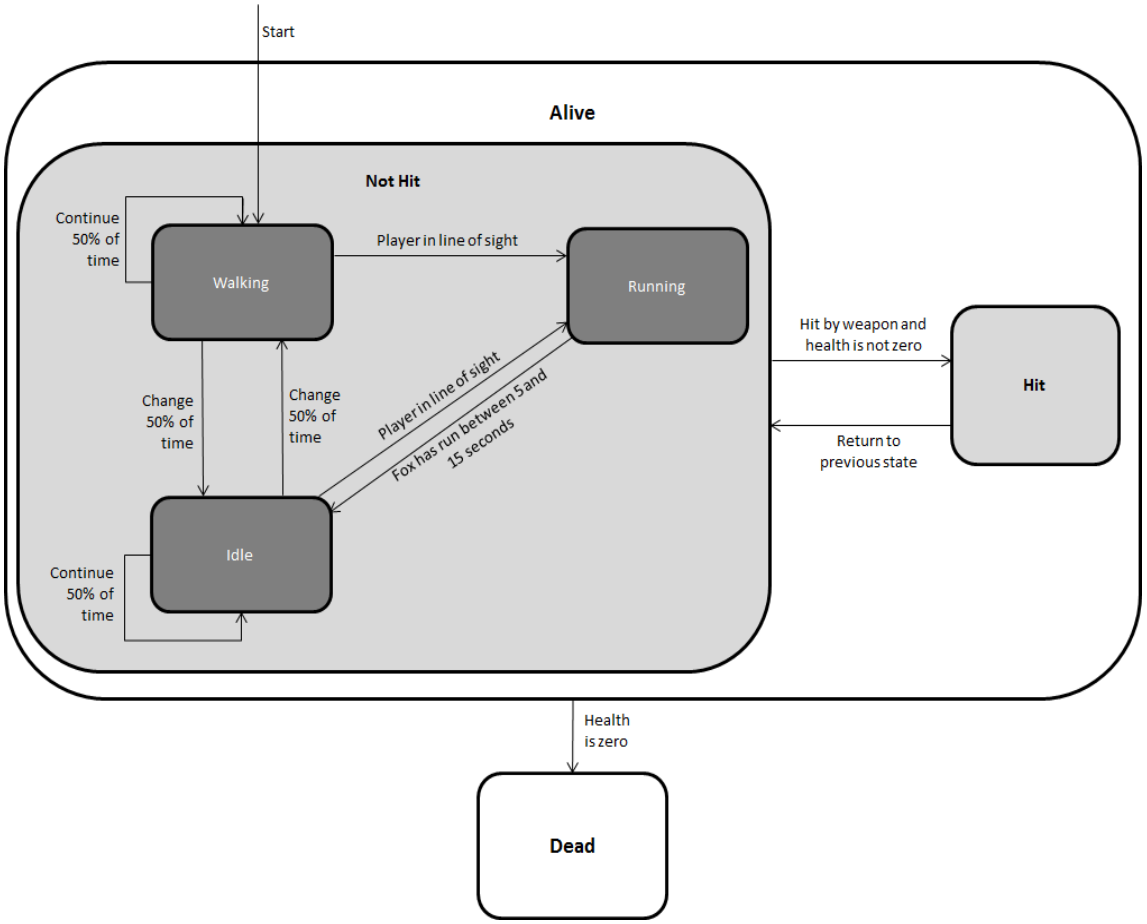
The stag is the final animal that can attack the player. The stag is different from the other animals, though, as it is meant to be more of a passive animal than an aggressive one. However, there is a slight chance that the stag will decide to attack the player instead of running away from them. When the stag does not see the player, it will randomly walk and go idle around the landscape. It also has a similar reaction to getting hit as the boar does, where it will turn to face the player before returning to its previous state.



Neutral Animals

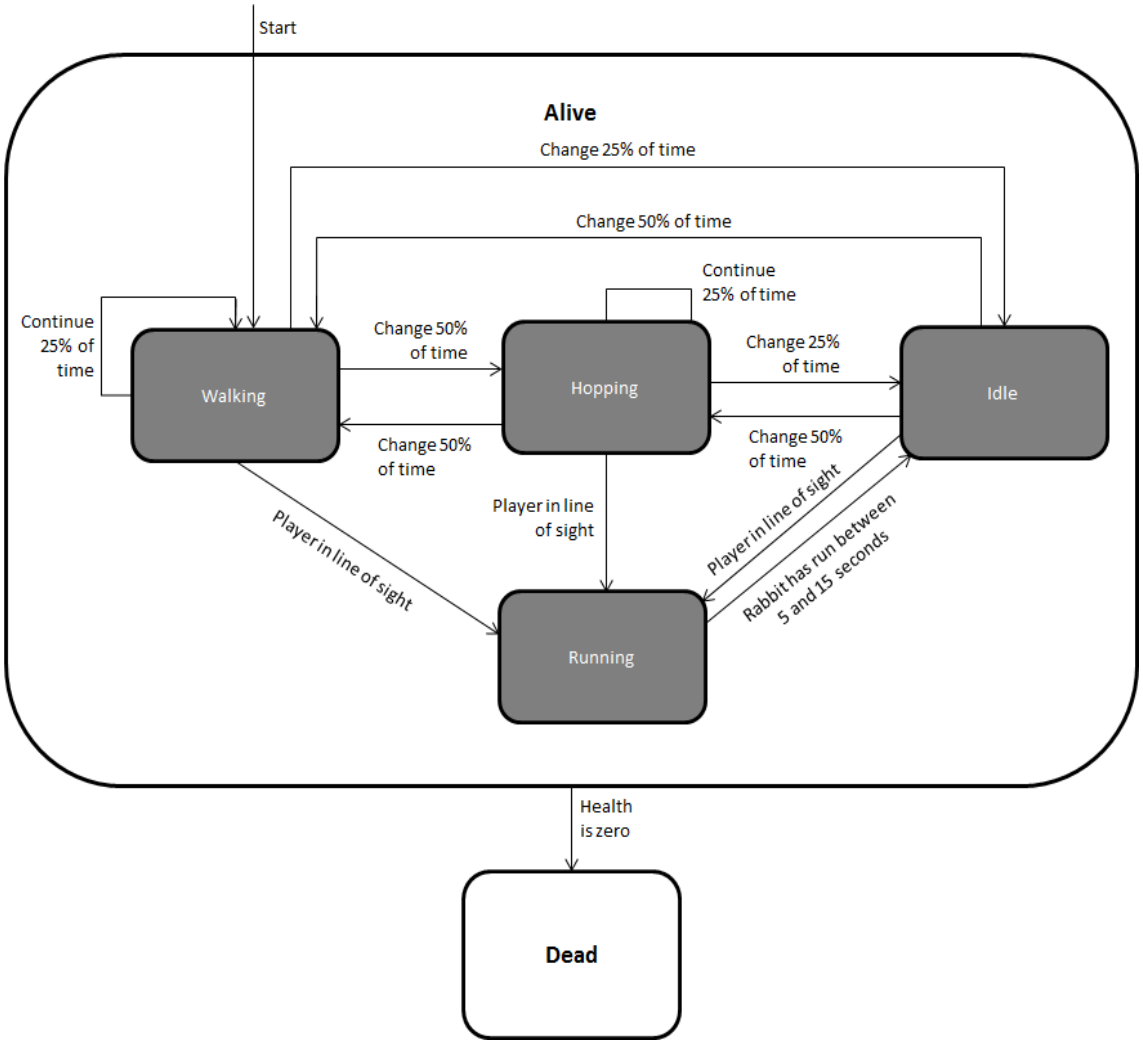
Fox

The fox is the first passive or neutral animal in the game. The fox is small and quick, making it hard for the player to spot them. The fox will mostly walk or idle when it doesn't see the player, but will run away from the player in the opposite direction that the player is at. The fox has an animation for when it gets hit by the player, but it will return to its previous state after being hit. The state diagram for the fox is much simpler compared to the enemy animals.



Rabbit

The rabbit is the other passive or neutral animal in the game. The rabbit is the smallest and quickest animal, so they are hard to find. The rabbit will randomly walk, hop, or go idle when it does not see the player. If it does see the player, it will run in the opposite direction from the player, then return to its random movement. The rabbit does not have an animation for being hit, so it will keep its state when hit by the player. The diagram below shows how the rabbit changes its states.



Environment

Landscape

The terrain is generated using many frequencies of Perlin Noise. By using multiple frequencies, the terrain has some large mountain areas but also some flat areas. The terrain is then textured based on the elevation and slope, so flatter sections are grass while the steeper areas are rocky. Environmental objects, like trees, grass, rocks, and other shrubs are placed manually into the game.

Sky

To make the game more realistic, a day and night cycle was implemented. The sun and the moon are both spheres that follow a circular path using Catmull-Rom spline cubic interpolation. Using cubic interpolation, control over the day and night can be easily maintained. The idea is to have the daytime gameplay last twice as long as the nighttime gameplay, that way the player can better fight and explore.

The sun and moon are directional lights that rotate as they move along their circular curve. This helps to simulate a full day transition by adding dusk and dawn characteristics to the scene. The light of the sun changes during the day, from yellow at dawn, to white at noon, then back to yellow at dusk. The moonlight is a pale blue and has a lower intensity than the sunlight as well.

The skybox also undergoes a transition from day to night. During the day, the backdrop is a blue color to simulate the daytime sky. During the transition to night, the skybox transitions or blends into the nighttime skybox, which is a field of stars. The transition from the nighttime to the daytime skybox occurs again at dawn.

Weather

Clouds

Dynamic clouds have also been added to increase the realism and complexity of the sky. There are five cloud coverage types: no clouds, few clouds, overcast clouds, heavy overcast clouds, and storm clouds. Clouds move from west to east in the terrain. All of the cloud types use the Unity particle system, but have different particle materials depending on the cloud type. Also, when it is night time, the cloud materials change to a nighttime material, which is visibly darker than the daytime cloud material. Here are the different cloud coverage types in effect during the game.

This is how the sky looks with no clouds in the sky. The sun is bright and you can see the nice blue color of the sky.

This is how the sky looks with the few clouds cloud type. The general cloud shape is the same for all clouds, but the size of them is different.

This is the sky with the overcast clouds in effect. Clouds are more plentiful than before and provide a more realistic look.

This is the sky with the heavy overcast clouds in effect. The clouds have almost filled the entire sky. Also, when these clouds are in effect, the sunlight intensity drops down a little bit in order to simulate the heavy cloud cover.

Finally, these are the storm clouds during a storm. The sunlight intensity is continuously lowered during a storm, simulating the effects of being caught in a severe rain storm. The clouds here do not have the same shape as the previous clouds, but show more light and dark patches in them. Storm clouds will replace the other cloud type that was previously in the sky, but they still start in the west and head east.
After a storm is finished, the next type of cloud cover is chosen to play. This choice is done randomly based on the four other cloud cover options, creating a more realistic environment through randomness.

Rain

A rain system was implemented to go along with the different cloud types in the game. This rain system helps to create rainfall depending on the initial cloud type in the sky before a storm arrives. If the few clouds cloud type was in the sky, then the rainfall that follows will be a light rainfall. If the previous cloud type was the overcast clouds, then the rainfall will be at a moderate level. If the previous clouds were the heavy overcast clouds, then the rainfall will be very heavy. The storm clouds will be in the sky before rainfall starts and stay in the sky during the storm.

The rain system uses a particle system that is attached the player, so the rain only falls around the player. This is implemented due to the high computational cost that comes with using a particle system to simulate rain. Because the rainfall happens around the player, the effect still works well and makes it seem like it is raining throughout the entire landscape.

As with most rain storms, there are flashes of lightning in the game. To simulate lightning, four directional lights are placed in the sky and are activated when the rain system is in use. The directional light that will be used to simulate the lightning is chosen randomly to make it seem like the lightning is not always coming from the same spot. These directional lights are light blue in color, similar to the moon light, but use linear interpolation to change the intensity of the lights when the lightning should be emitting. This gives the effect of lightning flashing on then off again very rapidly during a storm.

Sound

Background/Ambient

During some of the menus, music is played to set the mood for the player. In the game, there are different background sounds during the day and night. During the day, the player can hear a subtle sound of birds chirping. During the night, there is the sound of crickets. These sounds are meant to add some depth to the game and make the player feel like they are in a forest.

3D Effects

All other sounds in the game will be 3D sounds. These sounds will be tied to certain objects or areas, where the player will hear the sound differently depending on distance from the object and orientation towards the object. The use of 3D sound will add to the immersive feel that the player is supposed to have. There are footstep sounds for when the player is walking. When the rainfall is in effect, different sounds play for the different rain types depending on the intensity of the rain. Bears will roar at the player if the player gets too close to them, which will help the player to find the position of the bear. The weapons make sounds when they are used to attack and the wolves have a response sound for when they are hit by the player attacks. These are just a few of the sounds added into the game to make the playing experience seem as realistic as possible.