# - TREE HUGGERS -

## GAMEPLAY

Tree Huggers is a first-person survival and exploration game. The goal is to survive for as long as possible. The player must gather resources to fend off hunger, thirst, and predators, or they will perish as a result.

## SETTING

Tree Huggers takes place in a forest environment, surrounded by lush vegetation, mountains, and a variety of wildlife. The environment's design is focused on creating a realistic and immersive experience.

## PLAYER

You are an explorer who wakes up in the wilderness with limited resources. You can walk, run, jump, eat, and drink and are also proficient in crafting and combat.

## STATUS BARS

Keep your status bars up to stay alive

Red - Health
Green - Stamina
Yellow - Hunger
Blue - Thirst

## ITEMS

The items in the game are used to help the player survive. The crafting system allows the player to create items that are not found naturally.

Crafting Items


Stick


String


Coal

Survival Items


Water


Meat


Health


Torch

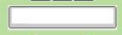
Combat Items


Bow


Knife

## CONTROLS

Can be rebinded in-game

Actions
- -Movement 
- -Jump 
- -Sprint 
- -Crouch 
- -Interact/Pick up objects 
- -Set waypoint marker 
- -Look (mouse)
- -Attack (mouse left click)

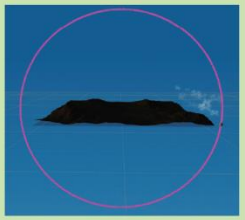
Menu
- -Enter/Exit map menu 
- -Enter/Exit menu 
- -Drop item (mouse middle click)

## DAY/NIGHT SEQUENCE

Catmull-Rom spline cubic interpolation was used to control the sun's/moon's trajectory and timing. Linear interpolation was used to alter the overall lighting of the scene. These created the day/night sequence when used in conjunction. Daytime is twice as long as nighttime to allow the player to fight and explore easier.

The pictures below showcase the day and night states.

The picture to the right shows the moon's/sun's curve based off the control points used in the Catmull-Rom spline.



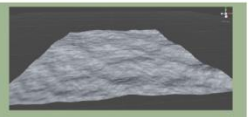
## WEATHER

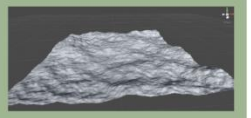
Tree Huggers incorporates a random, dynamic weather system to add realism to the environment. The clouds are an art asset that use a Particle/VertexLit Blended shader material to give the appearance of shaped clouds. Unity's particle system was then used to generate the clouds and the rain. Cloud coverage ranges between no coverage to heavy overcast. When the cloud system is in a heavy overcast state, cloud coverage increases and sunlight intensity decreases to mirror real-life behavior.

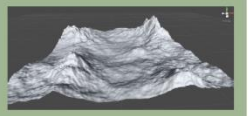
During a rainy state, the cloud particle system switches to emitting a storm cloud type and the rainfall type depends on the previous cloud coverage in the game. Rain only falls on the player to reduce computational cost and maintain a believable effect. During a stormy state, lightning may occur. Directional lights and sound effects were used to create this effect. Linear interpolation was used to vary the intensity of the light emmision, giving a sudden, flashing effect.

## TERRAIN GENERATION

Multiple Perlin Noise functions based off properties such as amplitude, frequency, and noise were used to generate the terrain. The pictures to the right show the terrain generation using one, two, and three Perlin Noise functions (top to bottom).







## ANIMALS/AI

Animals inhabit the world of Tree Huggers. Each animal uses a finite state machine to determine their actions and a sight system to detect the player. The animals will act accordingly to player actions. Animals have either enemy or neutral artificial intelligence, which is detailed below.

NEUTRAL


Nimble and small. Rabbits will run from the player if sighted.


Quick and wily. Foxes will run from player if sighted.

ENEMIES


Big and strong. Bears initially roar to scare player. Will attack if player decides to advance.

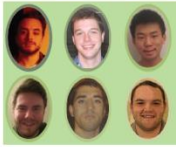

A mostly passive animal. Deers may attack player if feeling threatened.


The hunter. Wolves will attack players if sighted and will actively hunt them during the night.


Aggresive. Boars will attack player if in sight.

The pictures to the left showcase the AI's vision system. The white line is the distance check to the player while red area within the cone corresponds to the boar's field of vision. In the upper picture, despite being in range, the player is not within the vision cone, thus the boar will not detect the player. In the lower picture, the player is within the boar's field of view, so it will now attack the player.

## DEVELOPERS



Zach Ison, Kevin Miller, Darren Ng

Doug Smith, Matt Tatoczenko, Marshall Williams

unity

The Ohio State University

NP COMPLETE