Alexander Vaughn

Milestone 2

Preface:

Title - Goblin Wars.

Genre - Base Building, Resource Management, Strategy, Tower Defense, RTS. Goblin Wars draws inspiration from games such as: Minecraft, Factorio, Dyson Sphere Program, Bloons TD, Clash of Clans, Age of Empires, ARK, and others.

Platform - Windows OS.

Game Design

Strategy and Decision Making:

Tower Placement - One thing that makes goblin wars unique is that enemies can spawn from different locations on the map. This naturally creates a necessity for the player to decide where to place towers in a strategic way such that a tower can provide maximum coverage and get the most out of their resources.

Resource Allocation - Resources are hard to come by. Spending them on the correct types of towers and buildings at the right times throughout the game is critical to players' success.

Variety of Encounter – The map is procedurally generated so every game a player plays is different. There are different types of towers, which enriches the gameplay experience. Each tower has a unique attack style which allows the player to strategize which tower should be placed where and how each tower can synergize with another. Additionally, there are different types of enemies. Each has unique capabilities that require the player to determine how best to defend against each enemy type.

Opposition:

Story/Conflict - Goblin Wars is a classic journey that illustrates the constant battle that we all are a part of in everyday life. Light vs Dark (Man vs Nature). Will good prevail?

Enemies – The player must be stronger than those who threaten the light (Man vs Man).

Tension:

Doubt – The player will often wonder the following:

"Did I place that tower at the correct spot?"

"Should I have placed it a little to the left?"

"Is that tower going to fall?"

"Is that location defended well enough?"

"Will the Palace of Light fall?"

"There are too many of them!"

Scarcity – The player will run out of resources, not being able to defend certain locations on the map as much as they want to.

Objective and Gameplay:

The player will start the game at the center of the map, next to the Palace of Light. The objective of the game is to defend the Palace of Light at all costs. Should the Palace of Light fall to the invading goblins, the world will be engulfed in darkness and all hope will be lost.

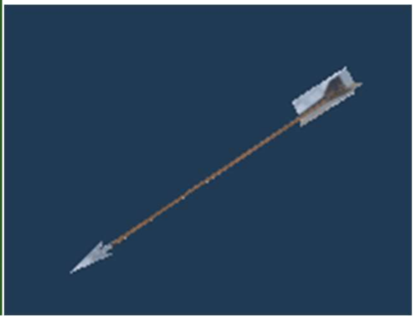
To prevent the fall of the light, the player will start the game with a few resources. These resources can be spent to build towers and purchase upgrades which will aid them in defending against the goblin assault. When a player feels ready, they may press the "start wave" button to begin spawning goblins.

If the goblins succeed in destroying the Palace of Light, the player loses the game. If the player manages to fend off all the goblins, they will complete the wave and be rewarded. When the player has cleared 30 waves of goblins, they will retreat. At this point the player is victorious, and they will be crowned as the Hero of the Light.

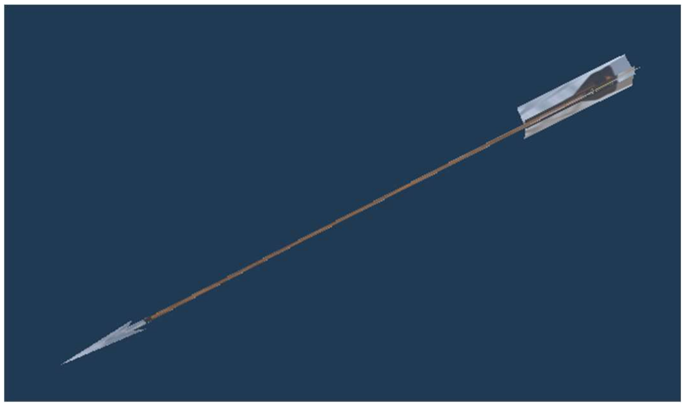
When a player completes a wave, they will be rewarded with resources and a new villager will be granted to them. The villager will spawn at the palace of light and automatically begin collecting resources. The resource the villager begins to collect will alternate every time one is spawned. This cannot be changed. Additionally, all buildings that have been damaged will be healed.

Core Units:

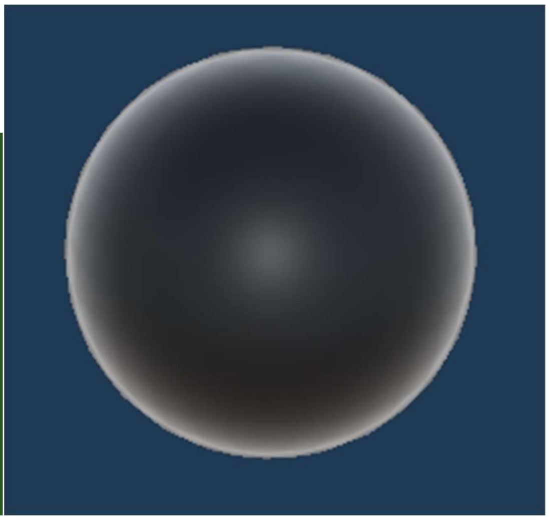
Archer Tower - The most affordable tower. Fires an arrow. Arrows do single target damage.



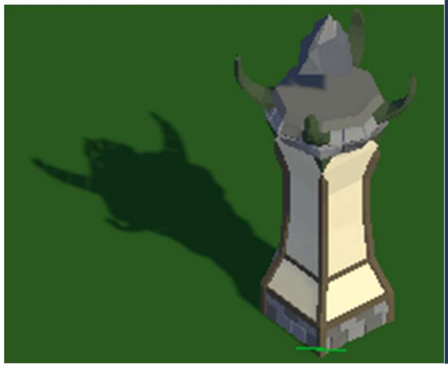
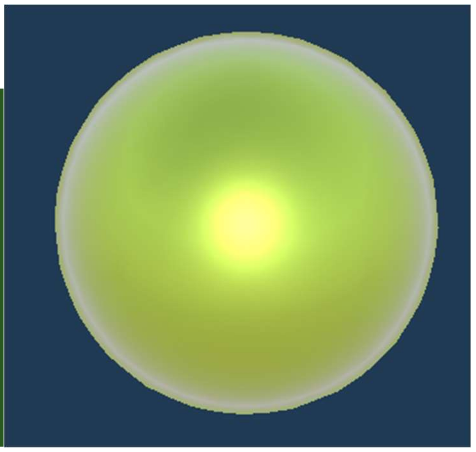
Ballista Tower - Fires a powerful ballista that deals high single target damage. Has a long attack cooldown.



Cannon Tower - Fires a cannon at high speeds. Does moderate single target damage. On impact, does area of effect damage to nearby goblins.



Poison Tower - Fires a poison bomb which travels slowly. On impact, infects enemies within an area. Enemies who are infected receive damage overtime.

Wizard Tower - Contains wizards who summon powerful meteors from the heavens. When a meteor impacts the earth, it deals a high amount of single target damage. Has a large blast radius which applies area of effect damage to nearby enemies. It also scorches enemies who are hit by the blast radius which deals damage to them over time. Has a long cooldown.

Green Goblin - A simple goblin grunt. Mostly used as cannon fodder by the goblin forces. Deals low single target damage and has low health but they spawn in numbers.

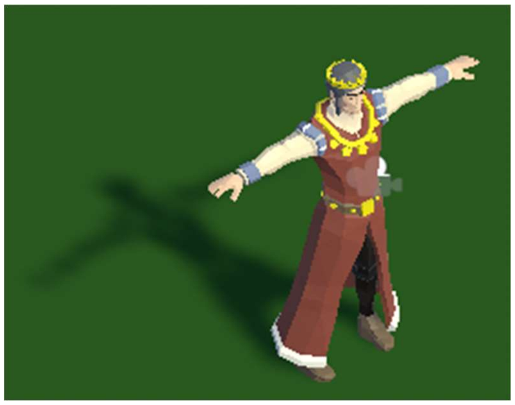
Blue Goblin – A powerful goblin soldier who deals a decent amount of single target damage and has a moderate amount of health.

Purple Goblin - Does moderate damage and has a moderate amount of health but moves very quickly.

Yellow Goblin – Has a moderate amount of health. Deals low single target damage but applies damage overtime to buildings.

Red Goblin - The most elite goblin soldier. These large brutes move slowly but deal high amounts of single target damage and area of effect damage to nearby buildings.

Player Character – The player in the game. Can move around and inspect how the base is holding up.



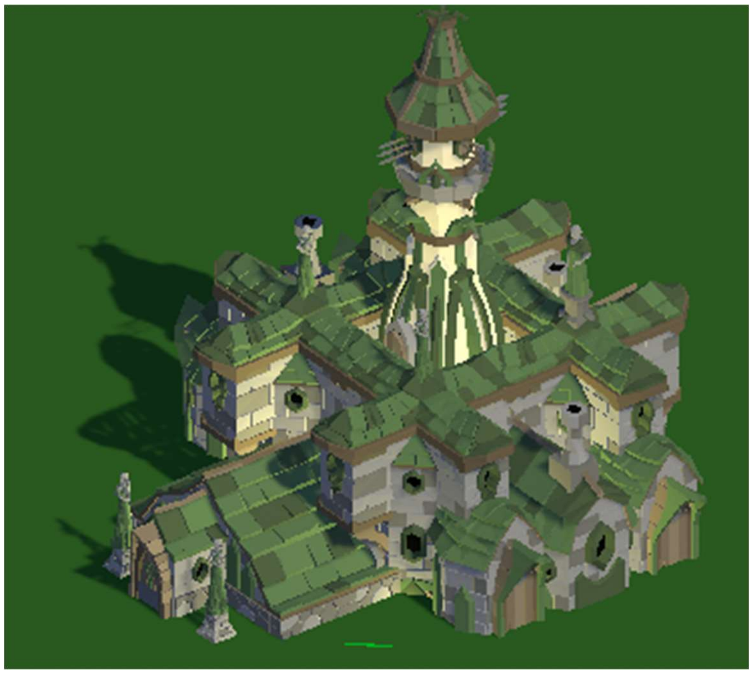
Villager – Collects resources from the world and brings them back to base.

Campfire – Pushes back the darkness and reveals more of the map in a radius.

Stone Camp – A drop off location for stone collected by villagers.

Wood Camp – A drop off location for wood collected by villagers.

Palace of Light – The core building in the game. It is being assaulted by the goblins. If this building is destroyed. The player loses the game. Also acts as a resource drop off point for villagers.



Play Testing

Play Test 1:

Tester 1 - I liked gathering resources and building towers and villagers. Defending your base with towers against waves of enemies was fun because it got harder as you played. But the game doesn't have a clear way to win, so it might get boring. You could add different goals, or maybe more complexity like different enemies, upgrades, or health for the nexus. You could also make resource gathering less predictable.

Tester 2 - Deciding where to station villagers to gather wood or food and how to allocate resources for building towers and villagers was fun and a strategic challenge. Adding ways for multiple players to play the game could increase its social appeal. Also, the scaling factor for the number of enemies should be adjusted, since at some point, there are just too many enemies.

How the play testing went - I had two different friends play test this game. Because my game is a 1 player game, I was able to have my friends play test independently. All I had them do was follow the Turn Steps and play the game. The feedback they provided was very useful. Especially that tips they made about changing the enemy scaling factor and things I could add to the game.

My Reflection - Going forward I am thinking about new ways I can scale up the enemies, add enemy types, add some durability to the nexus, add upgrades, make resource changes, and more.

Play Test 2:

Survey –

1. Please enter your name.

2. How intuitive did the controls feel?

Not Intuitive 1 2 3 4 5 Very Intuitive

3. How did the player movement feel?

Lethargic 1 2 3 4 5 Energetic

4. How did the enemy strength scaling feel?

Enemies are too weak 1 2 3 4 5 Enemies are too strong

5. How did the tower strength scaling feel?

Towers are too weak 1 2 3 4 5 Towers are too strong

6. How did the cost of the towers feel?

Towers are too cheap 1 2 3 4 5 Towers are too expensive

7. How did the cost of the upgrades feel?

Upgrades are too cheap 1 2 3 4 5 Upgrades are too expensive

8. Please enter any other comments here. What could be better? What could be added to the game?

Tester 1 - It could be difficult to implement at our level, but I think some sort of hit indicator for the arrows would be beneficial. It was difficult to tell if my arrows were hitting the goblins.

Tester 2 - I would like to see the basic attack (left click) to shoot slower but do more damage. Also, AI being able to path around instead of a straight line.

Tester 3 - F5 view mode, individual gold per enemy

Tester 4 - This was awesome! I think some improvements might be the random movement from the goblins, so they are not all in a line. Overall, it was awesome! I like the tower defense style and I like the Idea of being a soldier in a tower defense world!

Tester 5 - Indicators that the upgrades were received.

Feedback Summary - I got a lot of good feedback. I really liked having a survey that play testers could fill out. Being able to target specific things about the game and getting feedback about each one was very useful. I also got a lot of cool new ideas that I could add to my game. Overall, it seems like that scaling of tower damage was pretty good, but the number of goblins scale a little too fast early on. Players liked the concept of the game, which is great news.

How I am going to use the feedback - I think I am going to add everything that the play testers mentioned to the game. I want to make the goblins more interactive and more dynamic. I would like to make different type of goblins with different move speeds and bigger health bars. I also like the idea of giving gold per goblin kill instead of for defeating a wave.

Play Test 3:

Tester 1 - The game feels like it starts in the middle. Maybe start the player with enough resources for 1 or 2 towers, and fewer goblins.

Tester 2 - The game is a cool idea. Since you already have the day and night cycles, maybe you could add modifiers like maybe the enemies get more damage at night.

Tester 3 - I noticed the towers stay the same price. It might be interesting to increase the price slightly every time you buy one.

Tester 4 - When the goblins attack the palace of light, it's hard to see them sometimes. Maybe add an outline so you can see them when they are behind the building. Also, the hitbox on the palace of light seems a little off in some places.

Tester 5 - It's a little hard to tell where you are placing a building. Maybe make the grid more visible and highlight the tiles where the building is hovering.

What I'm going to change - I like the idea of making the grid and the goblins more visible with outlines. I will implement that. The modifiers idea is interesting. I might add that. I don't think I want to change the price of towers when you buy them, but it is something I will keep in mind. I will also change the starting resources to be lower and reduce the wave size at the beginning. I think this will give the player a better sense of progression in the game.

Play Test 4:

Tester 1 - The archer towers seem a little weak. Maybe give them more base damage, crit damage, or just scale their damage up as the waves get stronger.

Tester 2 – The drop off locations are a little buggy. The workers are walking past them sometimes. They should also stay around the drop off site when they acquire a new location to collect resources from.

Tester 3 – It's cool that the player character can move around in the world. It would be nice if they could also attack and help defend against the goblins.

Tester 4 – You already have the tool tip that shows up when you hover over a building. Maybe add a description to the tooltip for each building so that the player knows what it does.

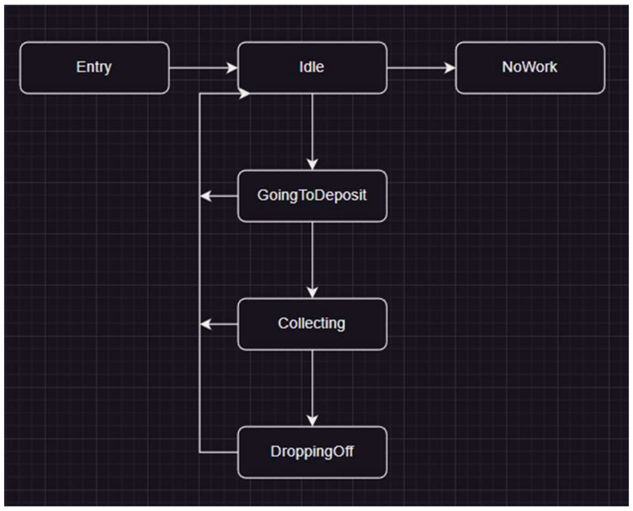
Tester 5 – Being able to command the villagers would be nice. That way you can choose what resource they are mining, and the player can decide what buildings they want and shoot for those.

What I am going to change - I think that I will add more information to the tooltip for each building. That seems like a simple thing that I could add without much work, and it would help the player immensely. I am going to fix the drop off locations, so they work better. I will make the archers towers stronger as well. I don't think I will change how the villagers work. I like where they are at right now. But I do like the ideas that were provided and I will continue to consider them in the future.

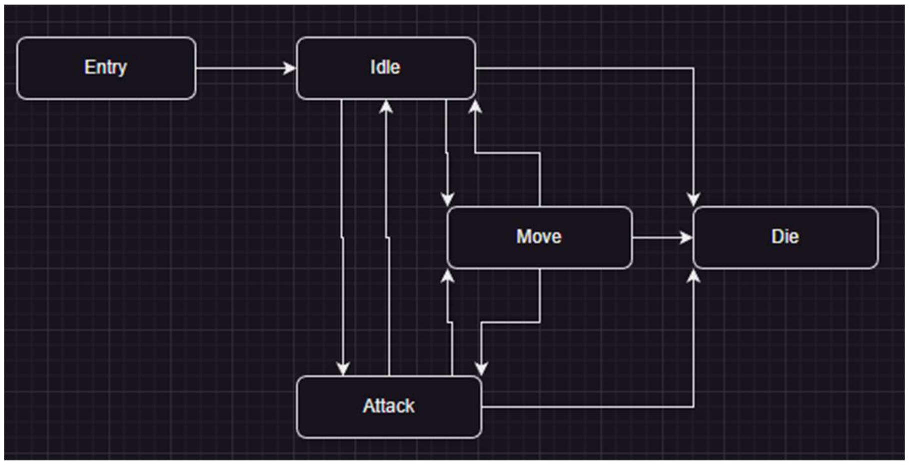
Game AI

State Machines**:**

Villagers - A state machine is used for the behavior of the villagers. The states are: Collecting, DroppingOff, GoingToDeposit, Idle, and NoWork. The NoWork state is a sink state. If this state is reached, there are no more resources on the map to be collected. When a villager is in this state it will not try to acquire any more resources. This will save computation.



Goblins - Another state machine is used for the behavior of goblins and their animations. The states are: Idle, Move, Attack, and Die. Once the die state is reached the Goblin object is destroyed.



Map Generation:

The map generation starts off by placing the palace of light at the center of the map. This must happen first so that a spot big enough is guaranteed to exist for this building. Subsequently stone deposits, then trees, will be generated.

Stone Deposits - There is a fixed number of stone deposits that will be generated for each map. For each stone deposit to be generated, an origin point is randomly selected on the map. If this origin point is a fixed distance away from any other stone origin point and there is nothing occupying the space at the origin, a new stone deposit is placed, and this origin point is saved in a list so that subsequent stone deposits will not be placed too close to another. This algorithm has the effect of spreading the stone deposits out quite evenly across the map.

Trees – Trees are spawned in clusters called forests. Just like the stone deposits, an origin point will be randomly selected for each forest. There is a fixed number of forests that will spawn. Each origin point will be remembered in a list so that they spawn evenly throughout the map. For each forest, a random radius we'll be chosen between two fixed points. This will give variety to the size of each forest. Then, the number of trees for this forest is pseudo-randomly selected.

The next step is to select a location for this tree within the forest. To do this, a random direction from the origin point is selected (360 degrees). Then a vector is created by multiplying the chosen radius by the square root of a random number between zero and one. This has the effect of having a higher density of trees closer to the center of the forest. This means that as the distance from the origin increases, there will be fewer trees. This creates a very natural looking forest.

It is important to note that although there are fixed numbers for the forest count and stone deposit count, these numbers cannot be guaranteed to be manifested in actuality, due to the randomness of the algorithm. This can be considered a feature as it adds to the randomness of the resulting map. For this reason, there is a maximum number of tries before the spawning attempt is halted.

Pathfinding:

Pathing - Pathing for entities in this engine was quite easy to implement with the use of Coroutines. Coroutines allow you to perform actions or a sequence of actions across multiple frames. During a wave, a Coroutine sends a goblin to a location. If it collides with anything during this process it will start a new temporary coroutine (stopping and attacking) before continuing. The pather component I created allows an object to be “commanded” to move to a specific location.

Software Design

Colliders - To get objects to interact with one another in the game world I gave a Collider and a rigid body to most objects in the game. This allowed me to use Unity’s event system and perform certain actions when an event occurred. An example of using this is with goblins. Whenever a goblin collides with a set of buildings that are whitelisted, the goblin will stop and attack that building. Once that building is no longer in the way it will continue its path.

Pather - The pather component works in tandem with the various Collider scripts. When a collision is detected then the pather component cancels the movement of the object. This script is responsible for starting a Coroutine and moving an object to a given location. It allows an object’s movement to be cancelled or continued.

Projectile - This component generates a projectile for a tower when it attacks. Projectiles use coroutines. When a projectile spawns, it is sent to a location. Once it reaches that location it times out and deals damage to a predefined entity. I thought about using colliders and the event system for this, but it seemed less reliable with moving targets, and more computationally expensive.

Targetable – The Targetable component’s primary purpose is to allow objects in the environment to damage other objects. The core pieces of this component include target acquisition, dealing damage to a target, setting the health of a target, and defining behavior for a target once it has died.

Goblin - The goblin component defines the behavior of the enemies in the game. It directs the goblin to attack a specific object once it runs into one or to ignore it. Goblins spawn in the fog of war and remain invisible until they've reached a lit section of the map. This script also specifies the animation to be played based on what the goblin is doing.

Tower - The primary role of the tower component is to acquire a goblin as a target and then repeatedly attack that goblin until it is dead. This component specifies the range in which a tower can acquire a target. It also specifies the projectile for a tower.

Bank - The bank component is a global Singleton. Its primary purpose is to manage the resources accumulated by the player. When a player purchases a new building, the bank will verify that the player has the required resources and then completes the transaction. It also maintains the costs of each object that can be purchased in the store.

Resource Collector - The resource collector component is what allows the villagers in the game to collect resources and drop them off at drop off sites. A simple call to unity's Physics.OverlapSphere allows the villagers to acquire the location of a new resource deposit or drop off site. A state machine is used to guide the villagers to their next action.

Wave Manger - The wave manager component simulates each wave. It maintains information about the current wave number and the difficulty of the next wave. This includes information about specific enemy types that should spawn, how many of them should spawn, and player rewards should the player beat the wave. It also locates new spawn points for the enemies throughout the wave and will switch up spawn locations every so often. Each time an enemy spawns, the likelihood that the spawn point will change increases. The enemy numbers and player rewards for each wave are calculated using mathematical functions. This makes it easy to recreate the information for a specific wave simply given the wave number.

Gui Manger - This component manages every UI element. It contains functions to show and hide certain UI elements. It also contains listeners for every button.

Placeable - This component allows a building to be placed on the map using the grid system. It specifies the grid size of a building. It also remembers the tiles that the building occupies. This is crucial in the building system, specifically, so that buildings cannot be placed on top of one another.

Ghost Build - When a player selects a building to be built in the UI and is subsequently verified by the bank, a ghost build object will be instantiated. A ghost build object is simply the building that has been chosen by the player but with most of its components disabled. While a ghost build object exists, each update cycle, it determines the world location below where the mouse is on the screen and moves the building to that location on the world grid. Once a player left clicks, they confirm that this is the location the building should be placed at, and the ghost build object is then destroyed . The new building is left in its place.

Attack Damage – The primary purpose for the attack damage script is to specify the behavior of an object when it attacks. There are a variety of attack types that a unit can use.

> The first is a single target attack which applies a constant amount of damage to a target.
>
> The second type of attack is Area of Effect. When a target is hit by an Area of Effect attack it takes a single target damage and then an additional damage amount will be dealt to neighboring enemies.
>
> The third type of attack is damage over time. When a target is hit by a damage over time attack it will take single target damage and then it will be inflicted with a modifier which applies an additional amount of damage to the target every set time interval for a set amount of time following the attack.

This component also keeps track of an attack cooldown for the object.

Map Generator & World Grid – As the game starts up, the map will be generated. The specific algorithms are discussed in more detail in the game AI section. The world grid creates a grid for the world using tile objects. The map generator generates trees, rocks, and the palace of light.

Light Emitter - When an object is instantiated with this component, darkness (or the fog of war) will be revealed within a certain radius of this object.

Implementation and Other Design Details

Game Engine:

I chose to use the unity engine for my game since I am most familiar with it. I've done many little personal projects with the unity engine in the past and so the systems are intuitive for me.

Perspective:

I chose to make the game with a 3D isometric perspective. I liked the idea of an isometric view because it feels very natural for top-down games. And since this is a cross between a tower defense and an RTS, it made the most sense.

I had the choice between making the game with a 2D isometric view or 3D isometric view. I ultimately chose the 3D isometric perspective because it was actually easier to work with since a lot of the assets I found for my game were 3D models. This also made it much easier to have shadows in the game and make it a little more immersive.

If I were to go with the 2D isometric view it would have been a lot harder to make the game look as good as it does. This is mostly due to the angle you must incorporate into the meshes of every object. With a 3D perspective you can simply adjust the camera angle and all those problems go away.

Unique Features:

Something that sets my tower defense game aside from others is that the player has a controllable character in the game, which allows the player to move around in the world and be more involved in defending the base. Normally in tower defense games, you watch your towers defend the base.

Another large difference is that the enemies can spawn from any direction. Normally, in tower defense games, enemies follow the same exact path. Goblin Wars takes a new and refreshing approach, which provides players with a more challenging and exciting experience.

One interesting thing that I did was make the map procedurally generated. I thought this would be a cool way to make every game unique and it would increase the playability.

I also added a little feature where there are day and night cycles in the world. I added this because I thought it would add to the theme of the game which is the battle between light and dark.

I used unity 's particle effects engine to make a fire effect. This effect was applied to both campfires which are light emitters in the game and to the fireballs that are cast by the wizard towers.