Cards of the Wild
Spring 2015 Battle Team
Final Individual Documentation

Elbert Dang

CSC 631-01

Dr. Ilmi Yoon

5/15/15

Battle Team Members	**Technical Group**
Nathanael Aff (Team Lead)	Server
Howard Aben	Server
Joseph Auby	Client
Kevin Qi	Client
Elbert Dang	Protocol
Dainius Grimalauskas	Database
Sonja Heikkinen	Art

My contributions to this project:

Although I was initially assigned to the Protocol Technical Team, our Client and Server team members found it much faster and simpler to work together directly to come up with the required protocols for our game, rather than go through me as a mediator. This was because everyone had already learned how to implement client/server protocols for our Project 1 at the start of the semester. Because of this, I focused my efforts on other tasks not related to Protocol, including:

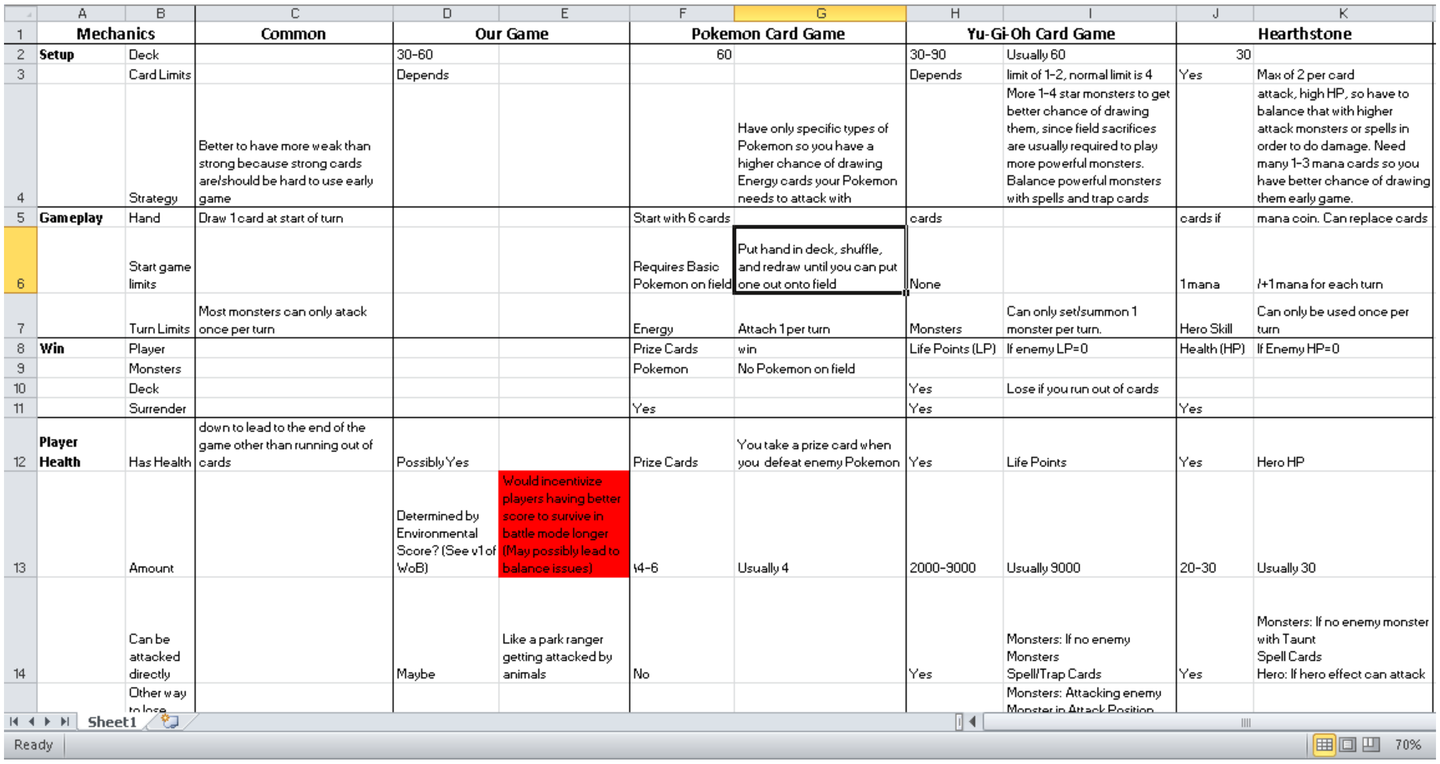
- **Game Concept Design**
  - Rules and Strategy
  - Database
    - Card Table
    - Deck as an array of card_ids
  - Other Ideas
    - Read Screen (BattleMainMenu)
    - Deck Editing (DeckEditMenu)
- **Art integration**
  - Initial Art Mockups
  - Prototypes
  - Procedurally generated cards
  - Prefabs
  - Shaders
- **UI/UX**
  - Hover Enlarge
  - Turn Indicator
  - Loading Screen
  - Game Over
- **Documentation**
  - Presentation slides
  - Google Drive for sharing files

Game Concept

Our group was initially set up as the Battle Team for the World of Balance game. However, given our limited experience, we decided to the easiest thing to implement was to have whatever battle we made turn-based. This proved to be a good decision, as another group that had a racing game ran into issues of lag and synchronization issues. The previous semester had partially implemented a battle arena-styled game, but our team leader, Nathanael, and several members of the Client team decided against parsing through last semester's code and believed it would be better to create our own game. We decided on a 2 player battle card game, who will play against each other using cards made from animals in their World of Balance environment.

I was able to come up with a lot of the game logic, basing it off of trading card games I had played in the past like Yu-Gi-Oh! and the Pokémon Trading Card Game. Several team members had also played Hearthstone, and since that game was more modern, all the members of our Battle Team decided to play it before our 1st milestone in order for us all to be familiar with how a trading card game worked. In order to create the rules for our own game, I started with doing research on the similarities and differences between Yu-Gi-Oh!, Pokémon, and Hearthstone, and began formulating what would work best for our own game (**https://docs.google.com/spreadsheets/d/1E6Zq7Qx9re7k_NAi4Ic-B4wiOPWG9SXgxryT7ZKilq4**).

Game Mechanics Comparison



	A	B	C	D	E	F	G	H	I	J	K
1	**Mechanics**		**Common**	**Our Game**		**Pokemon Card Game**		**Yu-Gi-Oh Card Game**		**Hearthstone**	
2	**Setup**	Deck		30-60		60		30-90	Usually 60	30	
3		Card Limits		Depends				Depends	limit of 1-2, normal limit is 4	Yes	Max of 2 per card
4		Strategy	Better to have more weak than strong because strong cards are/should be hard to use early game				Have only specific types of Pokemon so you have a higher chance of drawing Energy cards your Pokemon needs to attack with		More 1-4 star monsters to get better chance of drawing them, since field sacrifices are usually required to play more powerful monsters. Balance powerful monsters with spells and trap cards		attack, high HP, so have to balance that with higher attack monsters or spells in order to do damage. Need many 1-3 mana cards so you have better chance of drawing them early game.
5	**Gameplay**	Hand	Draw 1 card at start of turn			Start with 6 cards		cards		cards if	mana coin. Can replace cards
6		Start game limits				Requires Basic Pokemon on field	Put hand in deck, shuffle, and redraw until you can put one out onto field	None		1 mana	/+1 mana for each turn
7		Turn Limits	Most monsters can only atack once per turn			Energy	Attach 1 per turn	Monsters	Can only set/summon 1 monster per turn.	Hero Skill	Can only be used once per turn
8	**Win**	Player				Prize Cards	win	Life Points (LP)	If enemy LP=0	Health (HP)	If Enemy HP=0
9		Monsters				Pokemon	No Pokemon on field				
10		Deck						Yes	Lose if you run out of cards		
11		Surrender				Yes		Yes		Yes	
12	**Player Health**	Has Health	down to lead to the end of the game other than running out of cards	Possibly Yes		Prize Cards	You take a prize card when you defeat enemy Pokemon	Yes	Life Points	Yes	Hero HP
13		Amount		Determined by Environmental Score? (See v1 of WoB)	Would incentivize players having better score to survive in battle mode longer (May possibly lead to balance issues)	\4-6	Usually 4	2000-9000	Usually 9000	20-30	Usually 30
14		Can be attacked directly		Maybe	Like a park ranger getting attacked by animals	No		Yes	Monsters: If no enemy Monsters Spell/Trap Cards	Yes	Monsters: If no enemy monster with Taunt Spell Cards Hero: If hero effect can attack
		Other way to lose							Monsters: Attacking enemy Monster in Attack Position		

Sheet1

Ready

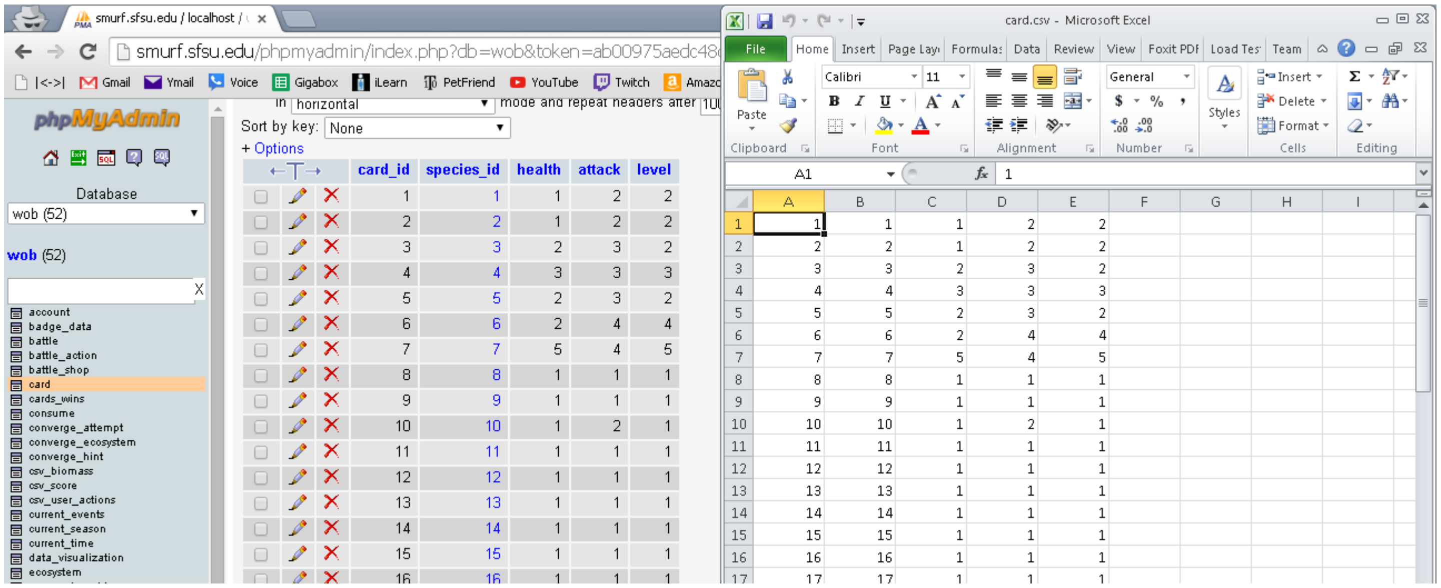
70%

Both players will each have a deck of cards, which are constructed from up to 30-40 of the animals within the World of Balance universe in addition to some game-specific spell cards. Currently, this deck is premade with a diverse array of cards that we have selected, although we would have liked to add the ability for the player to edit their deck and/or import animals from their Lobby Environment. During the battle, player Life Points in which the game ends when their Life Points reach 0. When battling, animals are summoned onto a playing field (max of 5) and can either attack the player to lower their Life Point, or other animals.

Sonja, our Art person, came up with the idea that the player should be represented by a Tree, and since I had noticed that the animals in the database already had their diet types categorized, we decided that the Herbivores can only attack the player's Tree of Life, Carnivores can only attack other animal cards, and Omnivores can attack both the Tree and other animals.

Each animal card will have a unique attack value that will be used to deal damage and a health value for receiving damage. Also, when an animal attacks another, they will be damaged by the amount of the defending animal's attack points. Omnivores could attack both animals and the Tree, so their stats would be lower than the others for balance reasons as well. Since we would be pulling the species names, diet_type and possibly their category (small animal, bird, etc) and descriptions from the database, we added a "card" table containing their attack, health, and level (which I filled in) and joined these two tables using each animal's species_id. Decks could also have been stored as an array of card_ids, but since that would been needed to be associated with each player, Nathanael found it easier to have it hard-coded server-side.

Uploading card information to database


smurf.sfsu.edu / localhost /
smurf.sfsu.edu/phpmyadmin/index.php?db=wob&token=ab00975aedc48
|<->| Gmail Ymail Voice Gigabox iLearn PetFriend YouTube Twitch Amazo
phpMyAdmin
Database
wob (52)
wob (52)
account
badge_data
battle
battle_action
battle_shop
card
cards_wins
consume
converge_attempt
converge_ecosystem
converge_hint
csv_biomass
csv_score
csv_user_actions
current_events
current_season
current_time
data_visualization
ecosystem
in horizontal
mode and repeat headers after 100
Sort by key: None
+ Options
card.csv - Microsoft Excel
File Home Insert Page Layout Formulas Data Review View Foxit PDF Load Test Team
Calibri 11 General
Paste Clipboard Font Alignment Number Styles Insert Delete Format Cells Editing
A1
fx 1

card_id	species_id	health	attack	level
1	1	1	2	2
2	2	1	2	2
3	3	2	3	2
4	4	3	3	3
5	5	2	3	2
6	6	2	4	4
7	7	5	4	5
8	8	1	1	1
9	9	1	1	1
10	10	1	2	1
11	11	1	1	1
12	12	1	1	1
13	13	1	1	1
14	14	1	1	1
15	15	1	1	1
16	16	1	1	1

	A	B	C	D	E	F	G	H	I
1	1	1	1	2	2				
2	2	2	1	2	2				
3	3	3	2	3	2				
4	4	4	3	3	3				
5	5	5	2	3	2				
6	6	6	2	4	4				
7	7	7	5	4	5				
8	8	8	1	1	1				
9	9	9	1	1	1				
10	10	10	1	2	1				
11	11	11	1	1	1				
12	12	12	1	1	1				
13	13	13	1	1	1				
14	14	14	1	1	1				
15	15	15	1	1	1				
16	16	16	1	1	1				
17	17	17	1	1	1				

At first, card levels were to be used similar to Yu-Gi-Oh's "sacrifice" mechanic in which more powerful cards had higher levels, and in order to summon higher leveled cards, a card or cards on the player's field would need to be removed on the field in order to play the high leveled card. Since only the lowest leveled cards can be brought onto the field without a sacrifice, more lower-leveled cards would be needed in the deck to get a higher chance of drawing them, or else the player would be defenseless. Since it made sense that prey animals were low-leveled and predators were higher, this also balanced the player's deck "environment" when there are more prey than predators. Also since players normally use low-leveled animals as sacrifices for higher-leveled ones, it would be akin to predators needing to eat prey for energy. For example, since there were "powerful" animals like the Lion, African Elephant, and Nile Crocodile, they would need 2 sacrifices from the field. Medium-sized animals would need 1, and smaller animals and insects would need none.

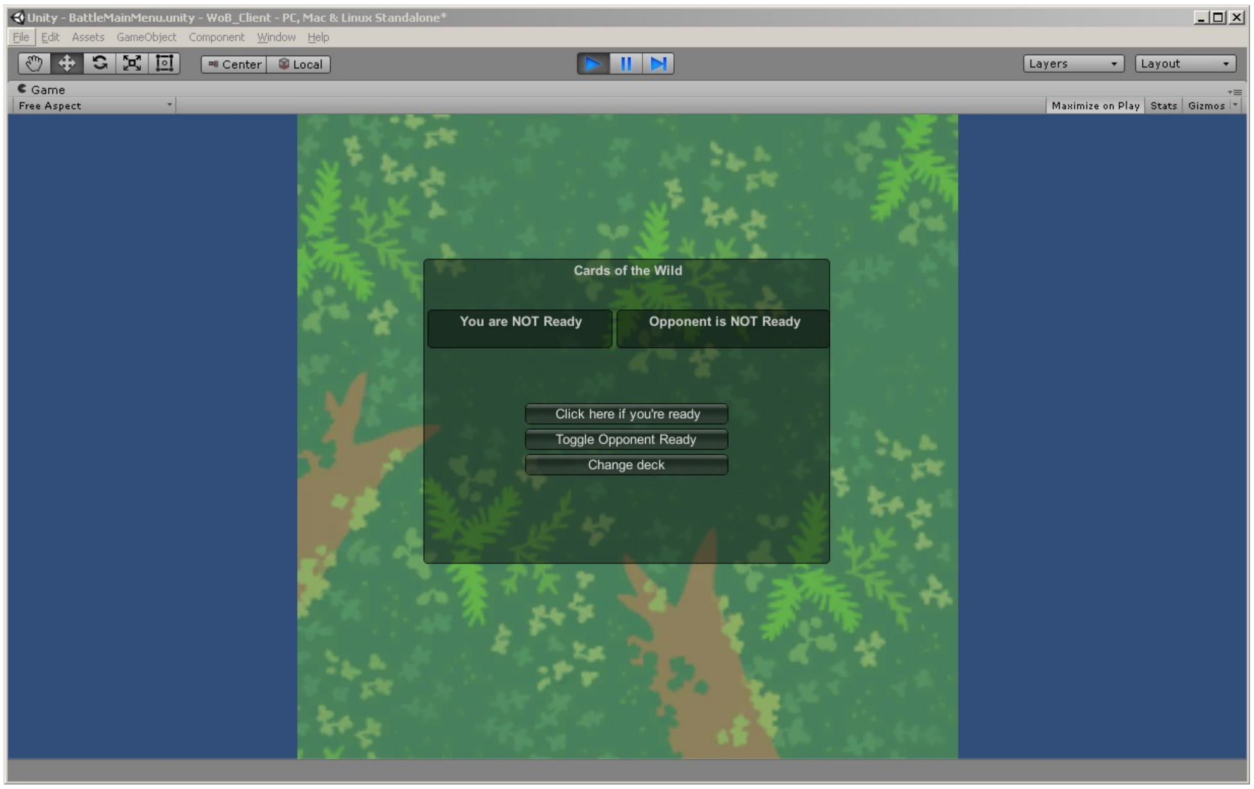
The sacrifice mechanic this proved to be difficult to implement however, so a simple resource system, similar to Hearthstone's mana, was used instead. In this system, players are given resources at the start of each turn, starting from 1 and gaining 1 maximum resource per turn, up to a final maximum of 9. All Cards would have a level, indicating how many resources would be needed in order to use them, so in general the more powerful the card, the more resources the player would need. Resources are refilled each turn, as well as increasing by 1 (up to 9). Either way, animals that were summoned could not attack the same turn, meaning it would need to wait a turn before it could attack. We had also wanted to implement spell cards with effects like environmental disasters which damage enemy animals or some that damage all animals on the field, buffs that temporarily or permanently increase one or more animals' attack or health, or those that affect the player like drawing extra cards from the deck, reviving a dead animal, or viewing the enemy player's hand, but didn't have enough time.

At the start of the game, a player is randomly determined to go 1st or 2nd and is given 3 cards from the top of their deck. At the beginning of each turn, players draw a card from their deck to their hand, up to a maximum of 7 in the hand. If the deck of cards is empty and there are no more cards to be drawn, the player loses. When attacking with animals on the field, the animal's attack points are subtracted from the target's health, and the target's attack is also subtracted from the attacker's health. If an animal's health reaches >=0, it "dies" and is removed from the field to the graveyard. After the player has finished with their turn, they end their turn and allow the enemy player to start their own turn. The game ends when a player's Tree of Life has >=0 health, when a players runs out of cards in their deck, or when a player surrenders and/or leaves the game. Both players will be awarded gold (with the winner getting more or course). There were plans to have it be a fixed amount with an extra bonus based on their lobby environmental score as a reward, but since we weren't sure if the Lobby would have that implemented, we gave 100 gold for the winner, 25 for the loser. This money would be used to purchase animals for the player's environment/card game and perhaps spell cards in the future.

Other ideas could include actual predator/prey relationships which may result in attacks doing more/less damage similar to Pokémon's attacks being super-effective with type advantages (water vs fire) or not-very-effective (water vs grass). Also, animals could have types which could be used in conjunction with spell cards. For example, a Pond could add defense to animals that could swim, like the Nile Crocodile and Catfish, but an earthquake could damage all animals, except ones who can fly like the African Grey Hornbill or Fish Eagle. Some animals could have effects of their own too, like maybe lowering the health of the African Elephant for enemy rodents on the field due to their musophobia (fear of mice), or maybe having the Black Mamba be able to "poison" a target, lowering the poisoned victim's health per turn. Also, if a DeckEditScene were to be implemented, an intermediate Ready Screen, or our BattleMainMenu, could have been implemented so that players can edit their decks before the battle starts, and press Ready when they are done. Once both players were finished and ready, the battle would begin.

Unimplemented Ready Screen

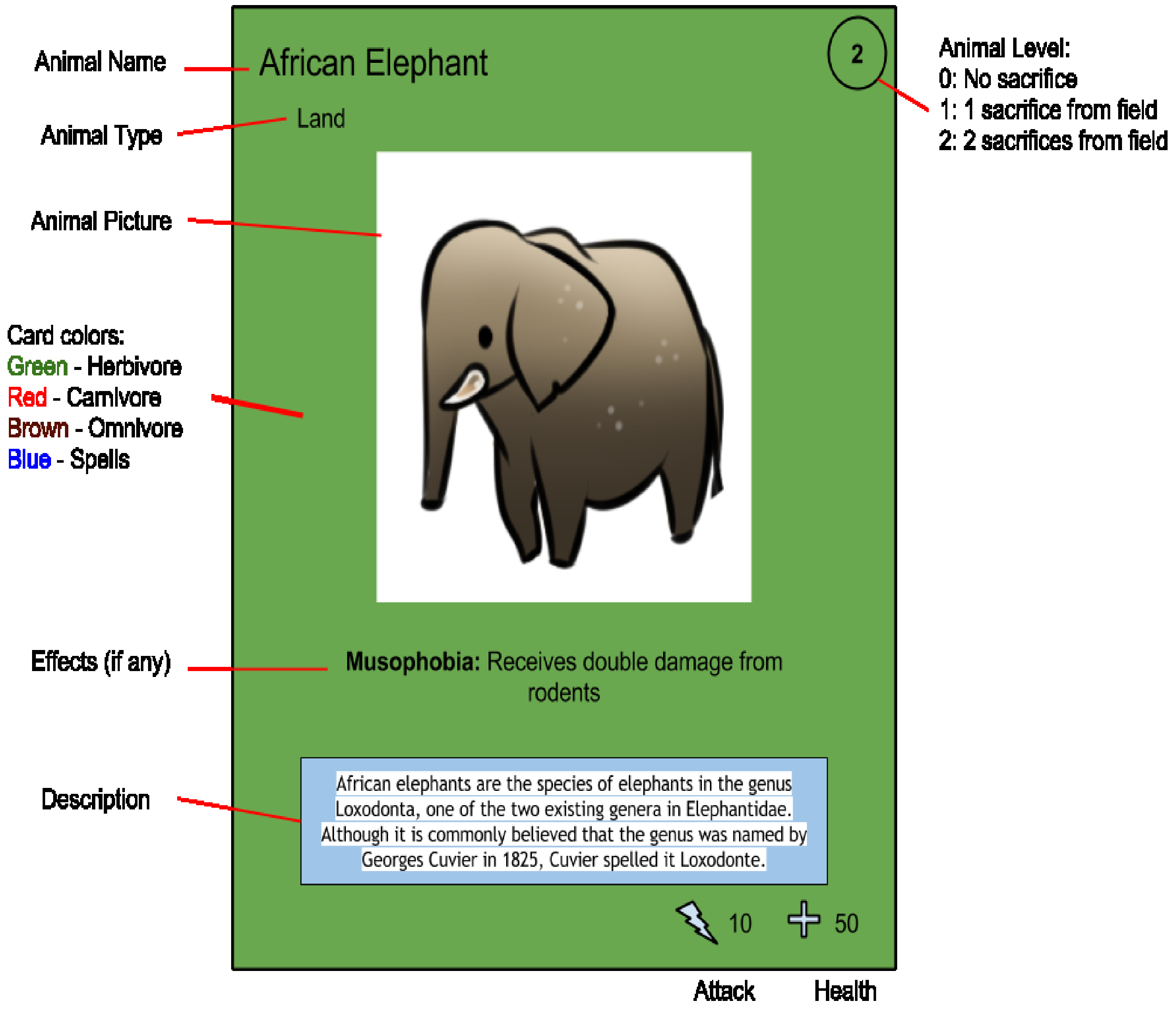
BattleMainMenu (with debug Toggle Opponent Ready Button)



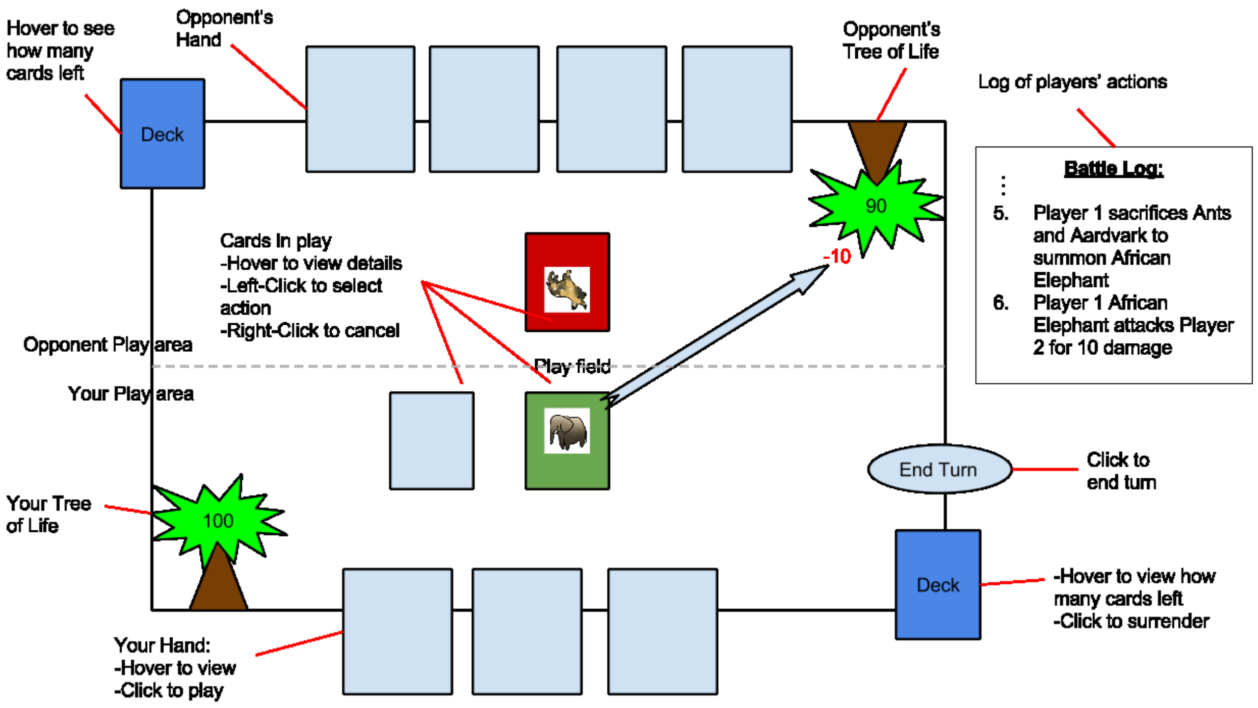
Art Integration

I came up with the initial UI Mockups and slides used for our Milestone Documentation Presentation.

Example Animal Card:


Animal Name
Animal Type
Animal Picture
Card colors:
Green - Herbivore
Red - Carnivore
Brown - Omnivore
Blue - Spells
Effects (if any)
Description
African Elephant
Land
2
Animal Level:
0: No sacrifice
1: 1 sacrifice from field
2: 2 sacrifices from field
Musophobia: Receives double damage from rodents
African elephants are the species of elephants in the genus Loxodonta, one of the two existing genera in Elephantidae. Although it is commonly believed that the genus was named by Georges Cuvier in 1825, Cuvier spelled it Loxodonte.
10
50
Attack
Health

Battle scene (Mirrored for each player):



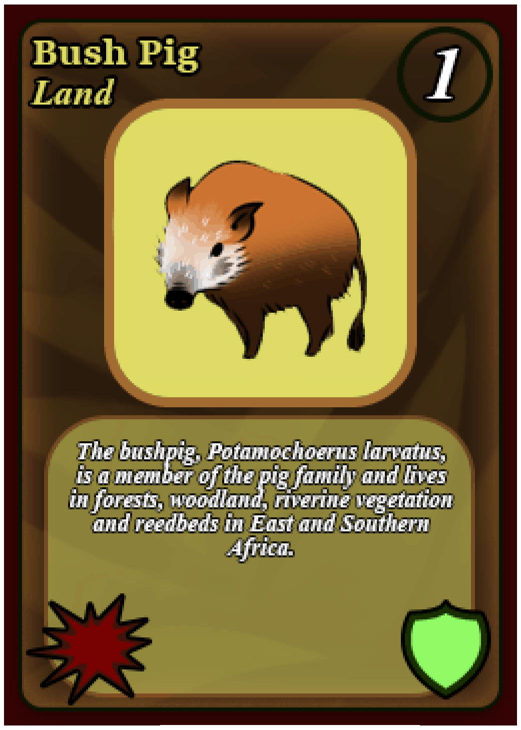
Battle Log:

⋮

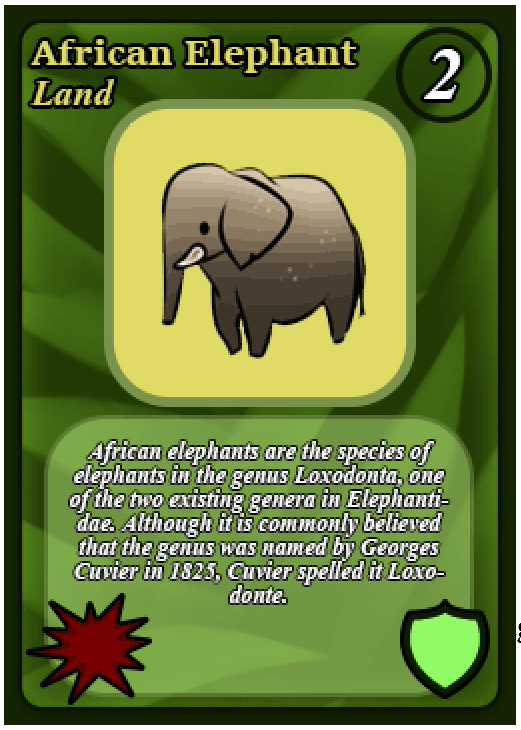
5. Player 1 sacrifices Ants and Aardvark to summon African Elephant
6. Player 1 African Elephant attacks Player 2 for 10 damage

This helped influence Sonja (our Art person) to make her own designs.

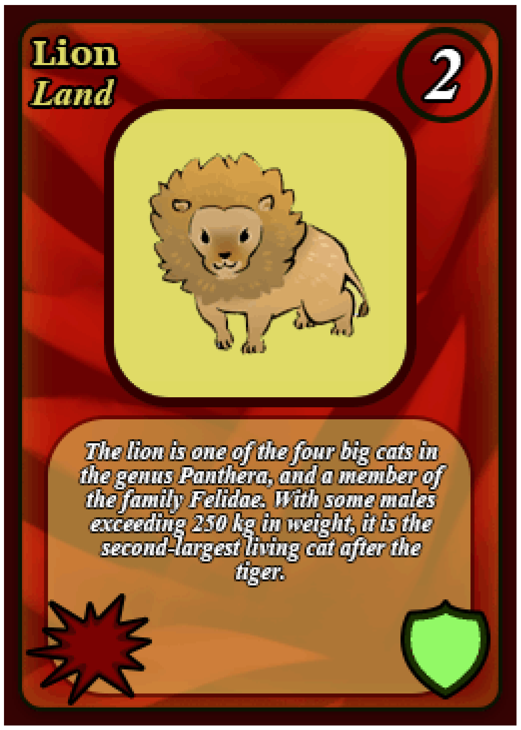
Updated Animal Cards


Bush Pig
Land
1
The bushpig, Potamochoerus larvatus, is a member of the pig family and lives in forests, woodland, riverine vegetation and reedbeds in East and Southern Africa.

Omnivore Card


African Elephant
Land
2
African elephants are the species of elephants in the genus Loxodonta, one of the two existing genera in Elephantidae. Although it is commonly believed that the genus was named by Georges Cuvier in 1825, Cuvier spelled it Loxodonte.

Herbivore Card


Lion
Land
2
The lion is one of the four big cats in the genus Panthera, and a member of the family Felidae. With some males exceeding 250 kg in weight, it is the second-largest living cat after the tiger.

Carnivore Card

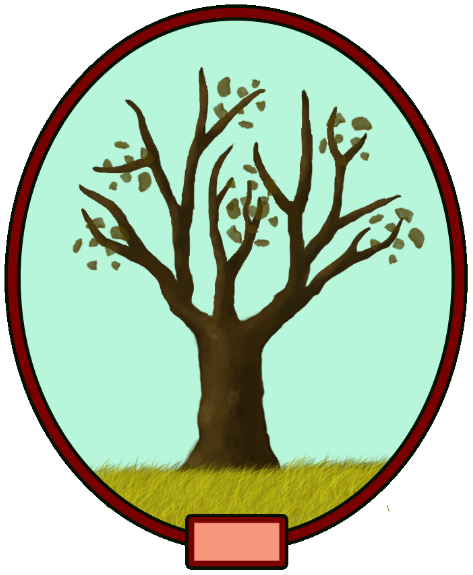
CARDS OF THE WILD

Back of cards/Deck

Tree(s) of Life

Healthy Tree

Dying Tree

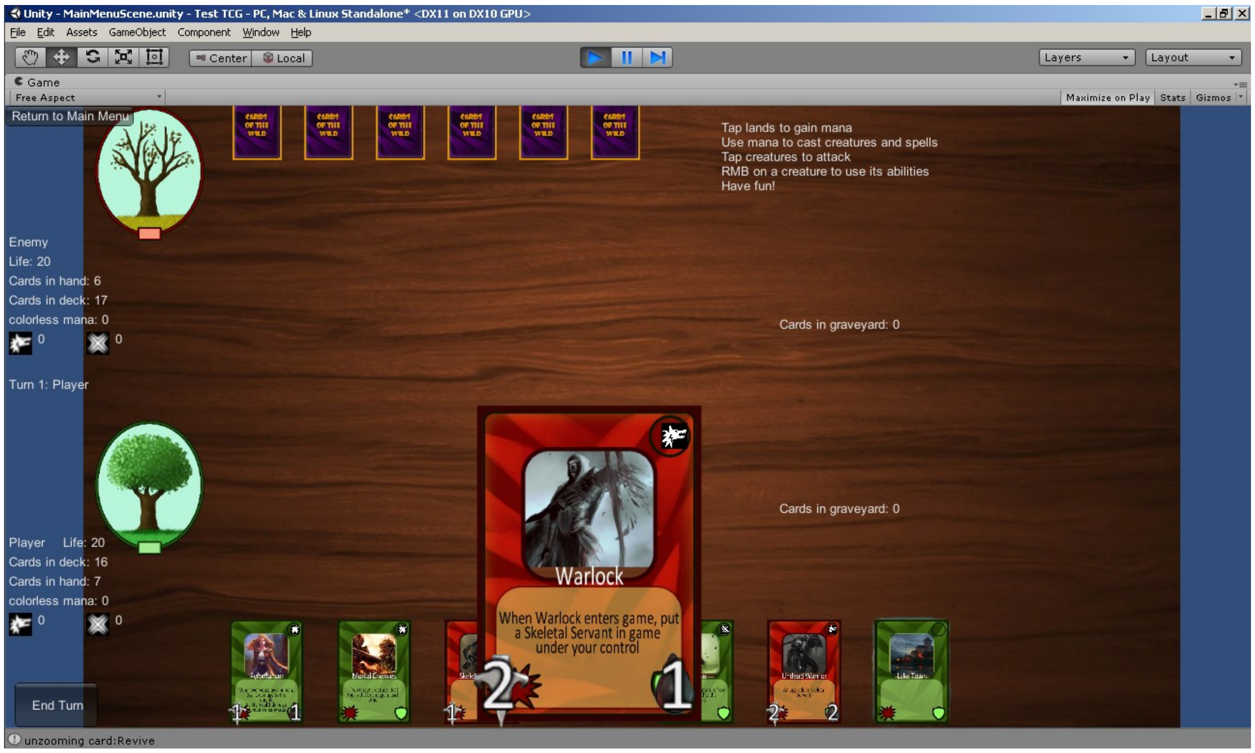


Dead Tree

Despite being the Team Lead for the Protocol Technical Team, I was not contacted by any other team for help with their protocols, so I decided to do even more research, this time on how to implement our game logic and art into Unity. I found that Hearthstone was created in Unity, and that someone had created a clone of it (http://forum.ragezone.com/f857/unity-3d-hearthstone-clone-1034061). However, I found that it did not work well as a resource due to it needing server calls, so I found a Unity Trading Card Game Maker (TCG) that we could use as a template instead (http://forum.unity3d.com/threads/trading-card-game-maker.237379). Using this, I was able to replace some art assets with our own to test some parts of our game, as well as see how they implemented their own game logic so I could help the Client team with some coding if needed.

Test Import into TCG template:

Playing field:


Unity - MainMenuScene.unity - Test TCG - PC, Mac & Linux Standalone* <DX11 on DX10 GPU>
File Edit Assets GameObject Component Window Help
Center Local
Layers Layout
Game
Free Aspect
Maximize on Play Stats Gizmos
Return to Main Menu
Warlock
End Turn

Tap lands to gain mana
Use mana to cast creatures and spells
Tap creatures to attack
RMB on a creature to use its abilities
Have fun!

Enemy
Life: 20
Cards in hand: 6
Cards in deck: 17
colorless mana: 0

Turn 1: Player

Cards in graveyard: 0

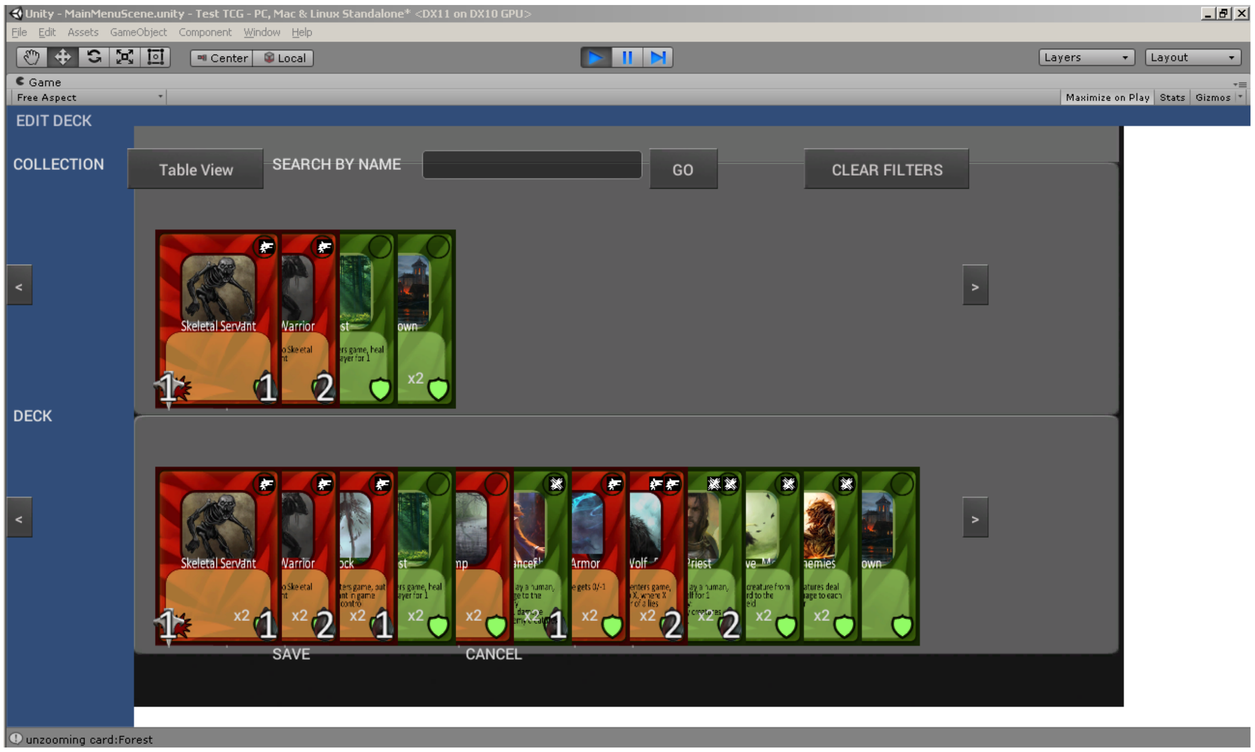
Cards in graveyard: 0

Player Life: 20
Cards in deck: 16
Cards in hand: 7
colorless mana: 0

When Warlock enters game, put a Skeletal Servant in game under your control

unzooming card:Revive

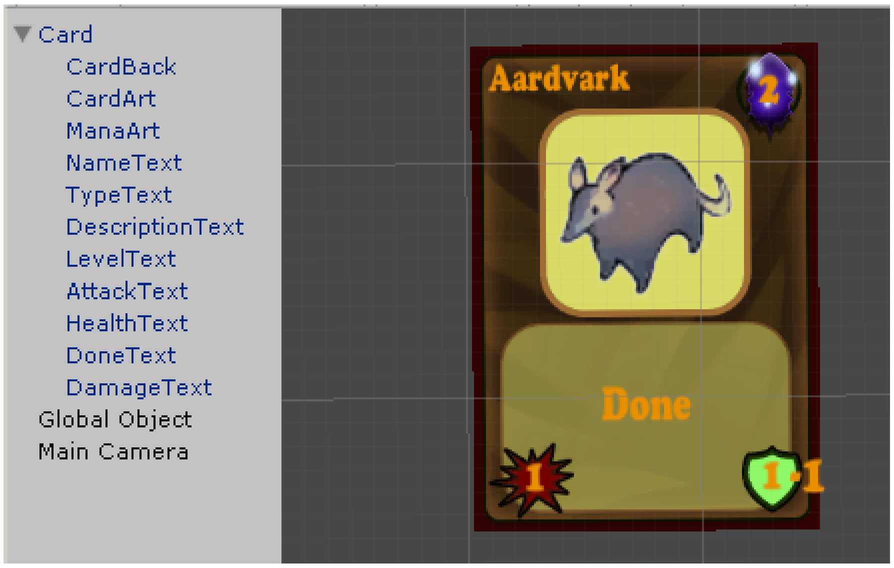
Deck Editor:



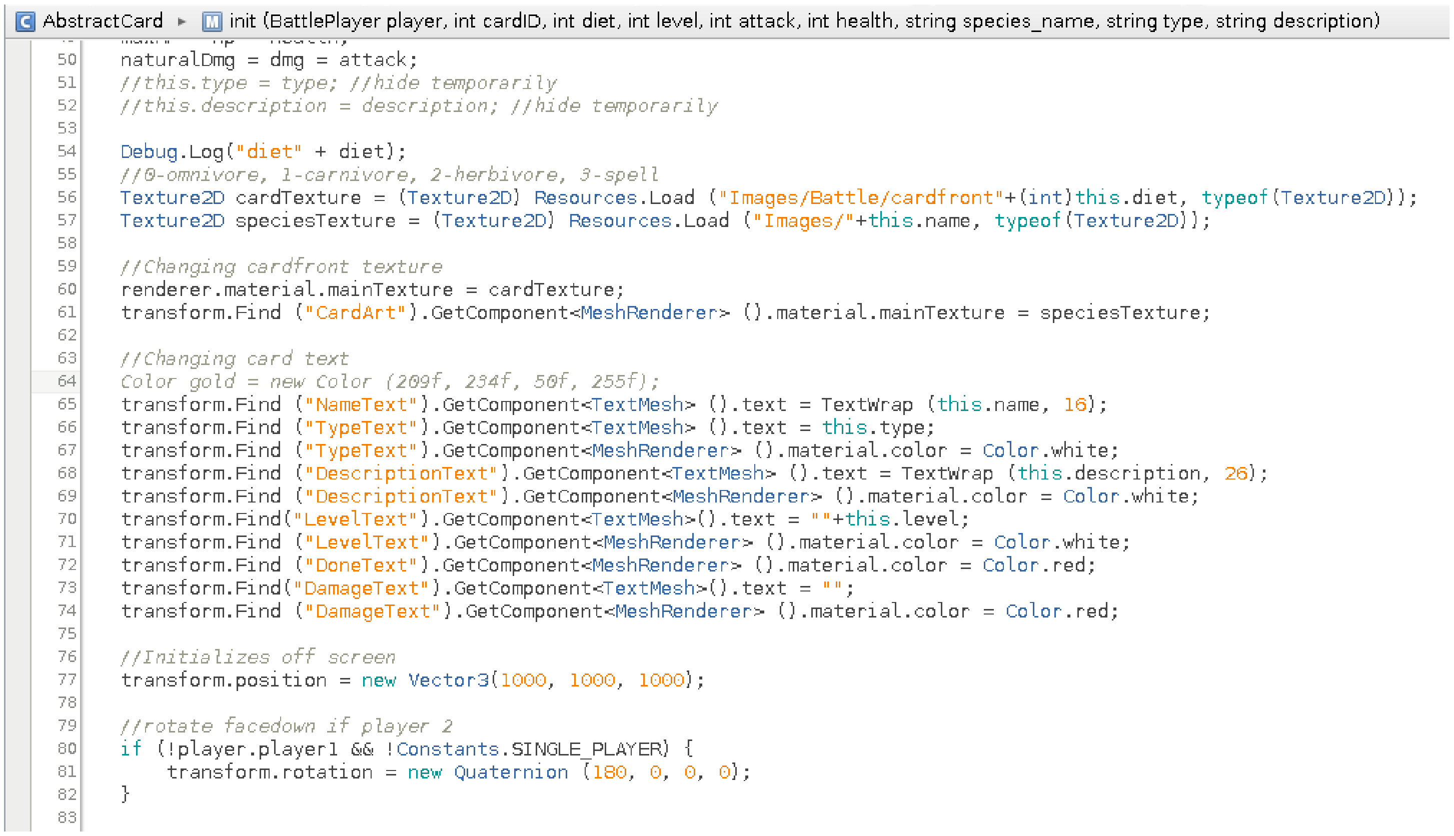
I noticed that the species table in the database had a “name” field matching the images in Resources/Images, we decided to pull (which I created a species table for).

Since I had previously noticed that the “name” in the species database table corresponded exactly to the image names of each animal in WoB_Client\Assets\Resources\Images, this led me to believe that cards could be procedurally generated. Therefore, I created a Card.prefab containing all the required information that each card would possibly need (for our game) and corresponding code to edit each card as it was instantiated.

Card Prefab



Card Instantiation

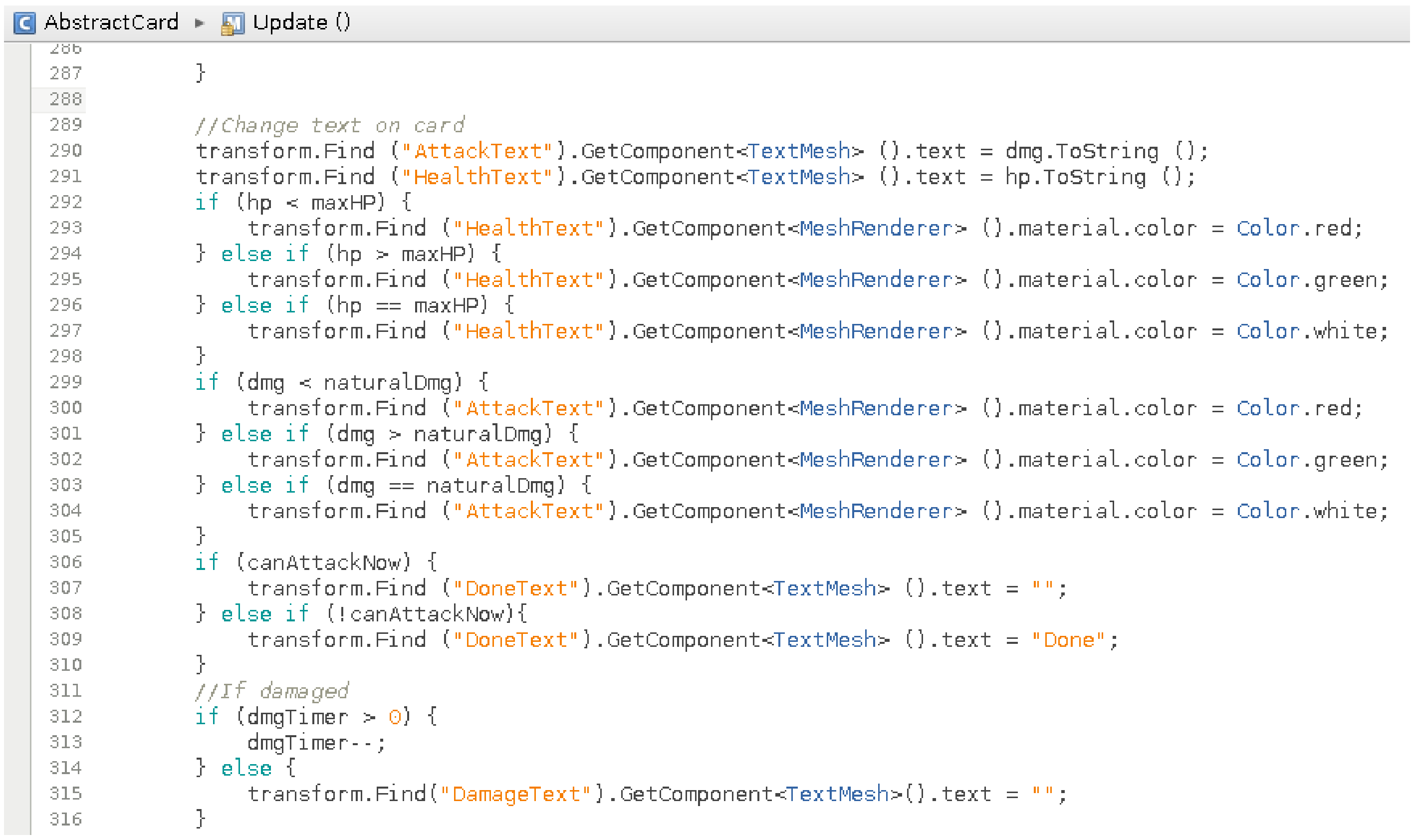


AbstractCard ▸ init (BattlePlayer player, int cardID, int diet, int level, int attack, int health, string species_name, string type, string description)

```
naturalDmg = dmg = attack;
//this.type = type; //hide temporarily
//this.description = description; //hide temporarily

Debug.Log("diet" + diet);
//0-omnivore, 1-carnivore, 2-herbivore, 3-spell
Texture2D cardTexture = (Texture2D) Resources.Load ("Images/Battle/cardfront"+(int)this.diet, typeof(Texture2D));
Texture2D speciesTexture = (Texture2D) Resources.Load ("Images/"+this.name, typeof(Texture2D));

//Changing cardfront texture
renderer.material.mainTexture = cardTexture;
transform.Find ("CardArt").GetComponent<MeshRenderer> ().material.mainTexture = speciesTexture;

//Changing card text
Color gold = new Color (209f, 234f, 50f, 255f);
transform.Find ("NameText").GetComponent<TextMesh> ().text = TextWrap (this.name, 16);
transform.Find ("TypeText").GetComponent<TextMesh> ().text = this.type;
transform.Find ("TypeText").GetComponent<MeshRenderer> ().material.color = Color.white;
transform.Find ("DescriptionText").GetComponent<TextMesh> ().text = TextWrap (this.description, 26);
transform.Find ("DescriptionText").GetComponent<MeshRenderer> ().material.color = Color.white;
transform.Find("LevelText").GetComponent<TextMesh>().text = ""+this.level;
transform.Find ("LevelText").GetComponent<MeshRenderer> ().material.color = Color.white;
transform.Find ("DoneText").GetComponent<MeshRenderer> ().material.color = Color.red;
transform.Find("DamageText").GetComponent<TextMesh>().text = "";
transform.Find ("DamageText").GetComponent<MeshRenderer> ().material.color = Color.red;

//Initializes off screen
transform.position = new Vector3(1000, 1000, 1000);

//rotate facedown if player 2
if (!player.player1 && !Constants.SINGLE_PLAYER) {
    transform.rotation = new Quaternion (180, 0, 0, 0);
}
```

Unfortunately, I found that every image in WoB_Client\Assets\Resources\Images had a white background instead of being transparent, making the cards look a bit ugly. Luckily I found a shader that would let us select a color (white in our case) and convert it to transparency (http://forum.unity3d.com/threads/cant-make-another-color-transparent.213407/). Also, since I wanted the Card.NameText to be visible no matter what colored background the front of the card was, and couldn't figure out how to change it to a gold/orange color that I wanted through code (since it wasn't a predefined Color), I made it the default color in the prefab and changed the other colors white and red as needed. After a card attacks or is summoned and cannot attack again, a "Done" text also appears. When the card receives damage, red DamageText also appears, but disappears after 2 seconds (120 frames).

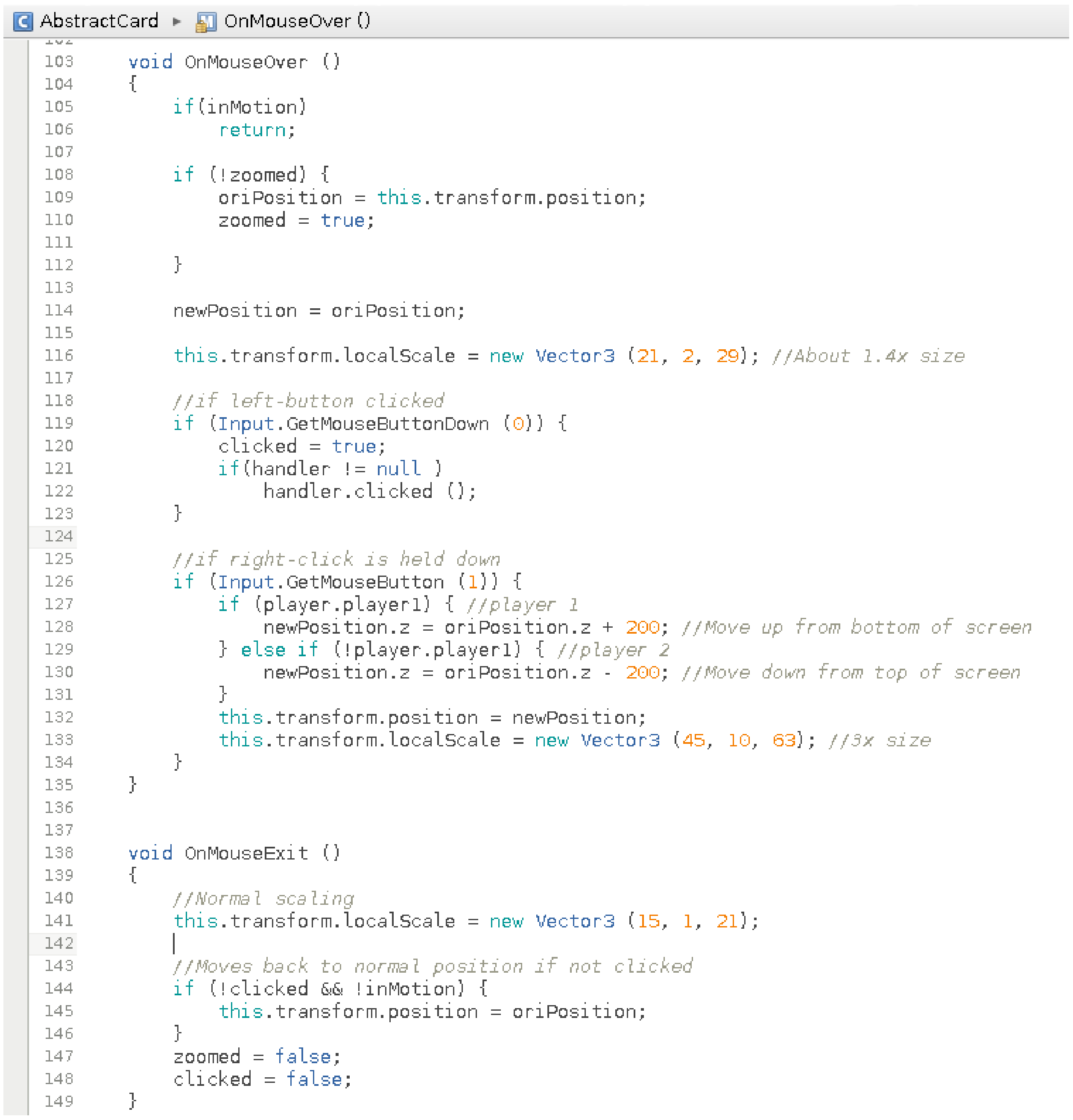
Updating Card Text



```
        }

        //Change text on card
        transform.Find ("AttackText").GetComponent<TextMesh> ().text = dmg.ToString ();
        transform.Find ("HealthText").GetComponent<TextMesh> ().text = hp.ToString ();
        if (hp < maxHP) {
            transform.Find ("HealthText").GetComponent<MeshRenderer> ().material.color = Color.red;
        } else if (hp > maxHP) {
            transform.Find ("HealthText").GetComponent<MeshRenderer> ().material.color = Color.green;
        } else if (hp == maxHP) {
            transform.Find ("HealthText").GetComponent<MeshRenderer> ().material.color = Color.white;
        }
        if (dmg < naturalDmg) {
            transform.Find ("AttackText").GetComponent<MeshRenderer> ().material.color = Color.red;
        } else if (dmg > naturalDmg) {
            transform.Find ("AttackText").GetComponent<MeshRenderer> ().material.color = Color.green;
        } else if (dmg == naturalDmg) {
            transform.Find ("AttackText").GetComponent<MeshRenderer> ().material.color = Color.white;
        }
        if (canAttackNow) {
            transform.Find ("DoneText").GetComponent<TextMesh> ().text = "";
        } else if (!canAttackNow){
            transform.Find ("DoneText").GetComponent<TextMesh> ().text = "Done";
        }
        //If damaged
        if (dmgTimer > 0) {
            dmgTimer--;
        } else {
            transform.Find("DamageText").GetComponent<TextMesh>().text = "";
        }
```

Since I noticed it was hard to tell whether or not clicking cards were actually selecting the correct cards or not, and since the card text was a bit hard to read, I added a zoom-enlarging visual effect when the mouse was hovered over them. Unfortunately I couldn't zoom into the card too much without messing up the card's location (because of the in-hand position code at the time), so I made the hover-zoom very slight and an optional large zoom when holding down the right mouse button.

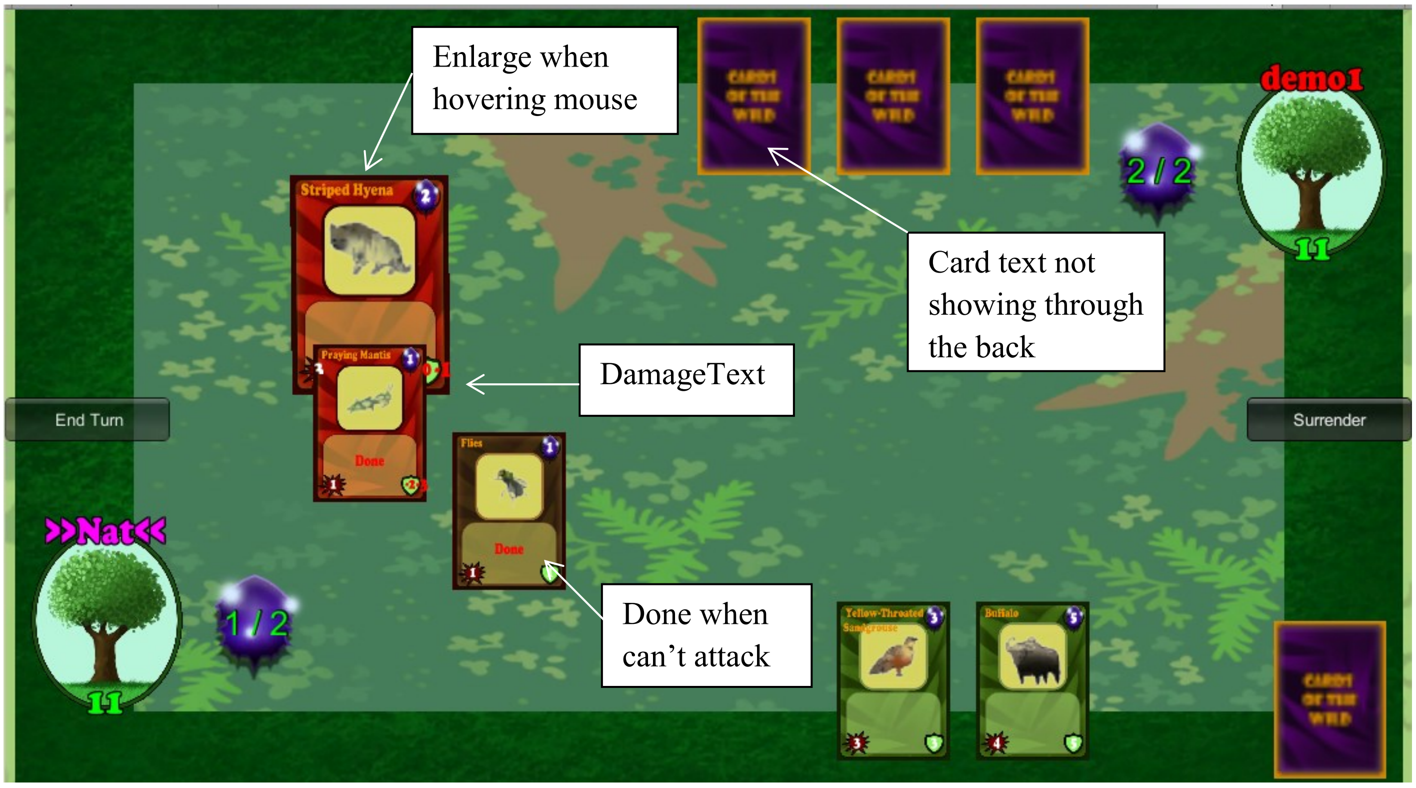
Card Zoom Enlarge



```
    void OnMouseOver ()
    {
        if(inMotion)
            return;

        if (!zoomed) {
            oriPosition = this.transform.position;
            zoomed = true;

        }

        newPosition = oriPosition;

        this.transform.localScale = new Vector3 (21, 2, 29); //About 1.4x size

        //if left-button clicked
        if (Input.GetMouseButtonDown (0)) {
            clicked = true;
            if(handler != null )
                handler.clicked ();
        }

        //if right-click is held down
        if (Input.GetMouseButton (1)) {
            if (player.player1) { //player 1
                newPosition.z = oriPosition.z + 200; //Move up from bottom of screen
            } else if (!player.player1) { //player 2
                newPosition.z = oriPosition.z - 200; //Move down from top of screen
            }
            this.transform.position = newPosition;
            this.transform.localScale = new Vector3 (45, 10, 63); //3x size
        }
    }


    void OnMouseExit ()
    {
        //Normal scaling
        this.transform.localScale = new Vector3 (15, 1, 21);

        //Moves back to normal position if not clicked
        if (!clicked && !inMotion) {
            this.transform.position = oriPosition;
        }
        zoomed = false;
        clicked = false;
    }
```

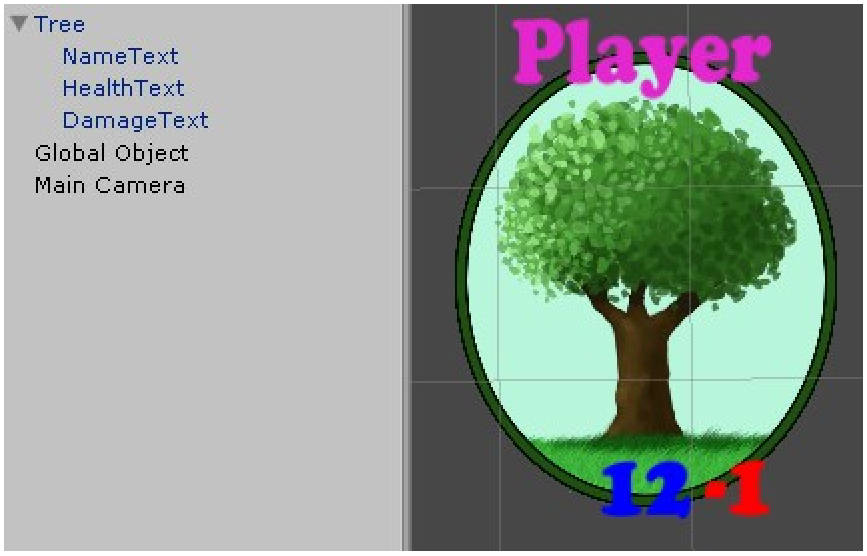
I also found a script to outline each letter of text with a colored outline (http://answers.unity3d.com/questions/542646/3d-text-strokeoutline.html) for better visibility, especially when zoomed out. When flipping the card over to show its back however, I found the text would display through our texture no matter what, so I found a shader that would make the text only viewable from the front (http://wiki.unity3d.com/index.php?title=3DText) and changed it accordingly to support 3D rotation. We found that the 3DText Shader and TextOutline script interfered with each other though, so we removed the TextOutline from the cards and instead added it to the Tree text since the tree would not be flipped.

All together

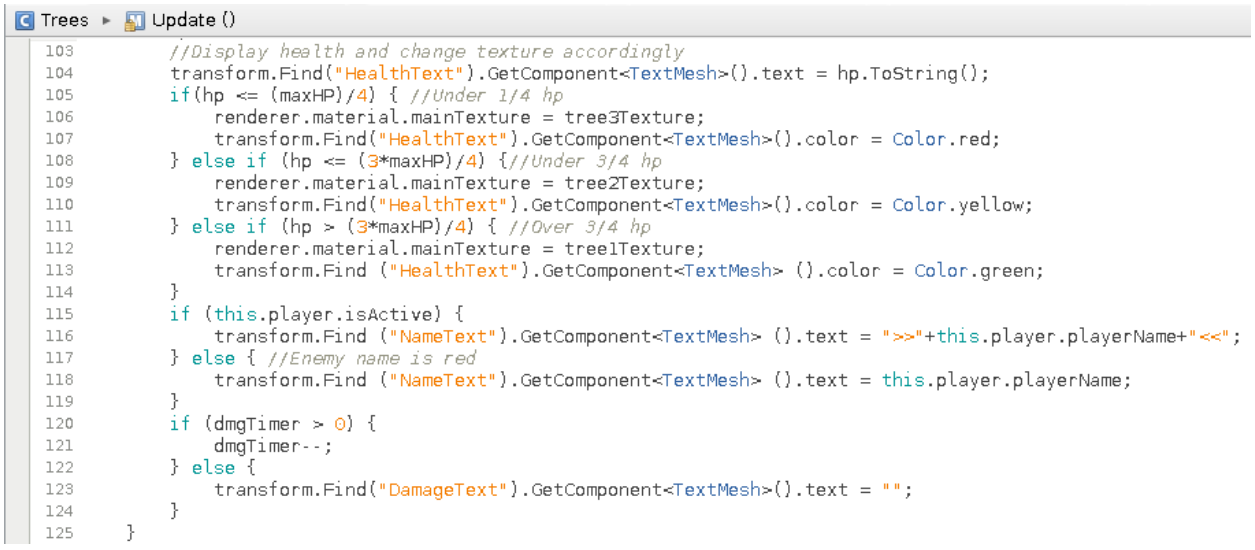


For the Player Trees, each tree would have the name of the player with a turn indicator on the player's name (in addition to there being a popup in the center of the screen). The tree also had the same slight hover-to-enlarge as the cards did, as well as DamageText showing when it was attacked, for clarity. In addition, I changed the Tree texture from healthy to dying, then to dead once its health reached certain levels.

Tree Prefab

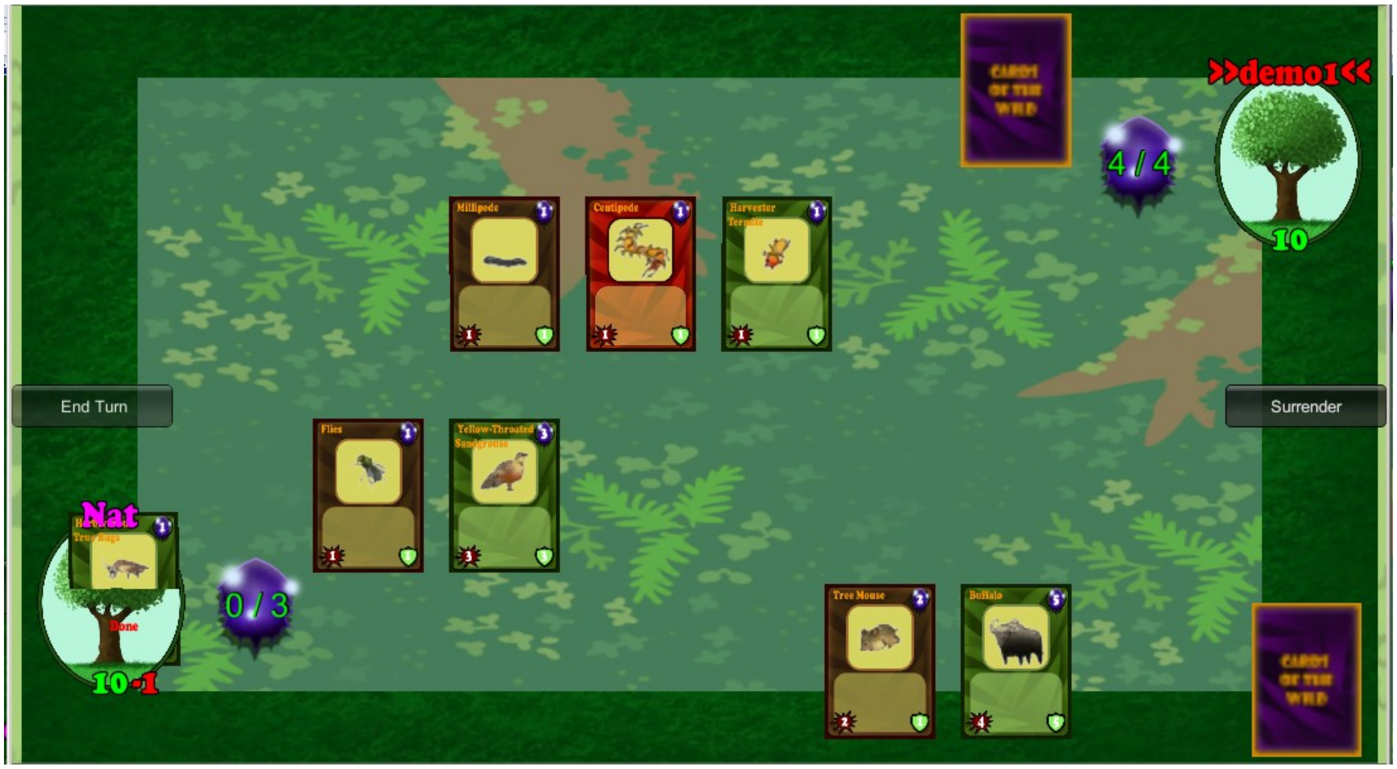


Updating Tree Textures/Text



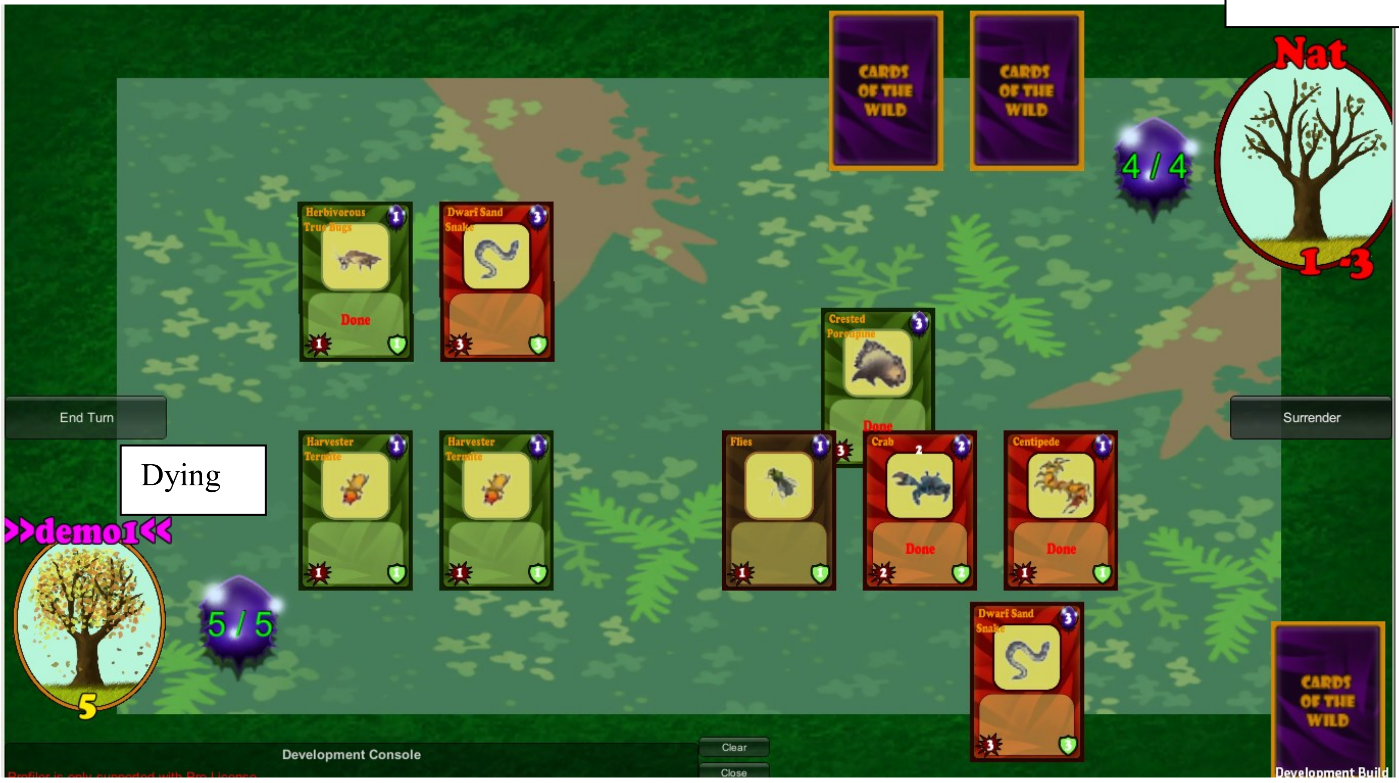
```
        //Display health and change texture accordingly
        transform.Find("HealthText").GetComponent<TextMesh>().text = hp.ToString();
        if(hp <= (maxHP)/4) { //Under 1/4 hp
            renderer.material.mainTexture = tree3Texture;
            transform.Find("HealthText").GetComponent<TextMesh>().color = Color.red;
        } else if (hp <= (3*maxHP)/4) {//Under 3/4 hp
            renderer.material.mainTexture = tree2Texture;
            transform.Find("HealthText").GetComponent<TextMesh>().color = Color.yellow;
        } else if (hp > (3*maxHP)/4) { //Over 3/4 hp
            renderer.material.mainTexture = tree1Texture;
            transform.Find ("HealthText").GetComponent<TextMesh> ().color = Color.green;
        }
        if (this.player.isActive) {
            transform.Find ("NameText").GetComponent<TextMesh> ().text = ">>"+this.player.playerName+"<<";
        } else { //Enemy name is red
            transform.Find ("NameText").GetComponent<TextMesh> ().text = this.player.playerName;
        }
        if (dmgTimer > 0) {
            dmgTimer--;
        } else {
            transform.Find("DamageText").GetComponent<TextMesh>().text = "";
        }
    }
```

Tree being attacked



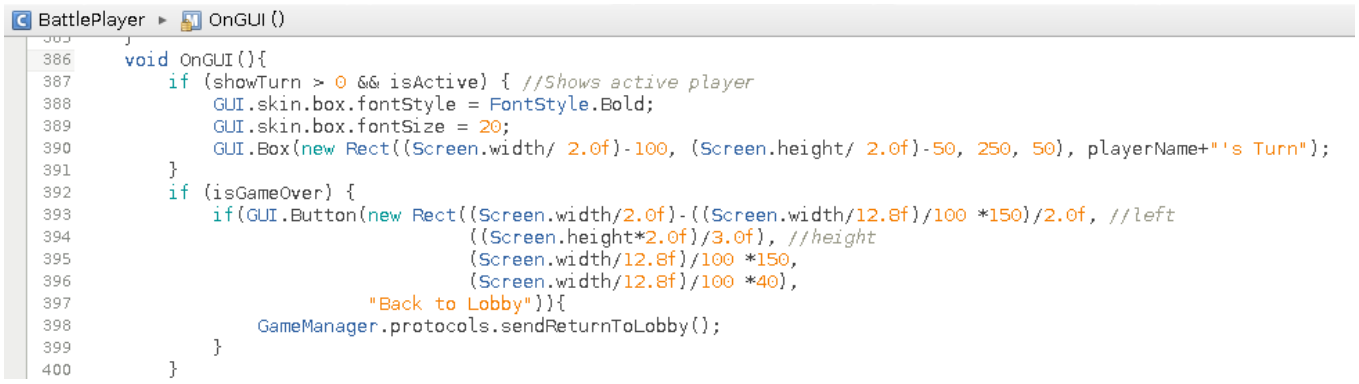
In-Game change of textures

Almost Dead



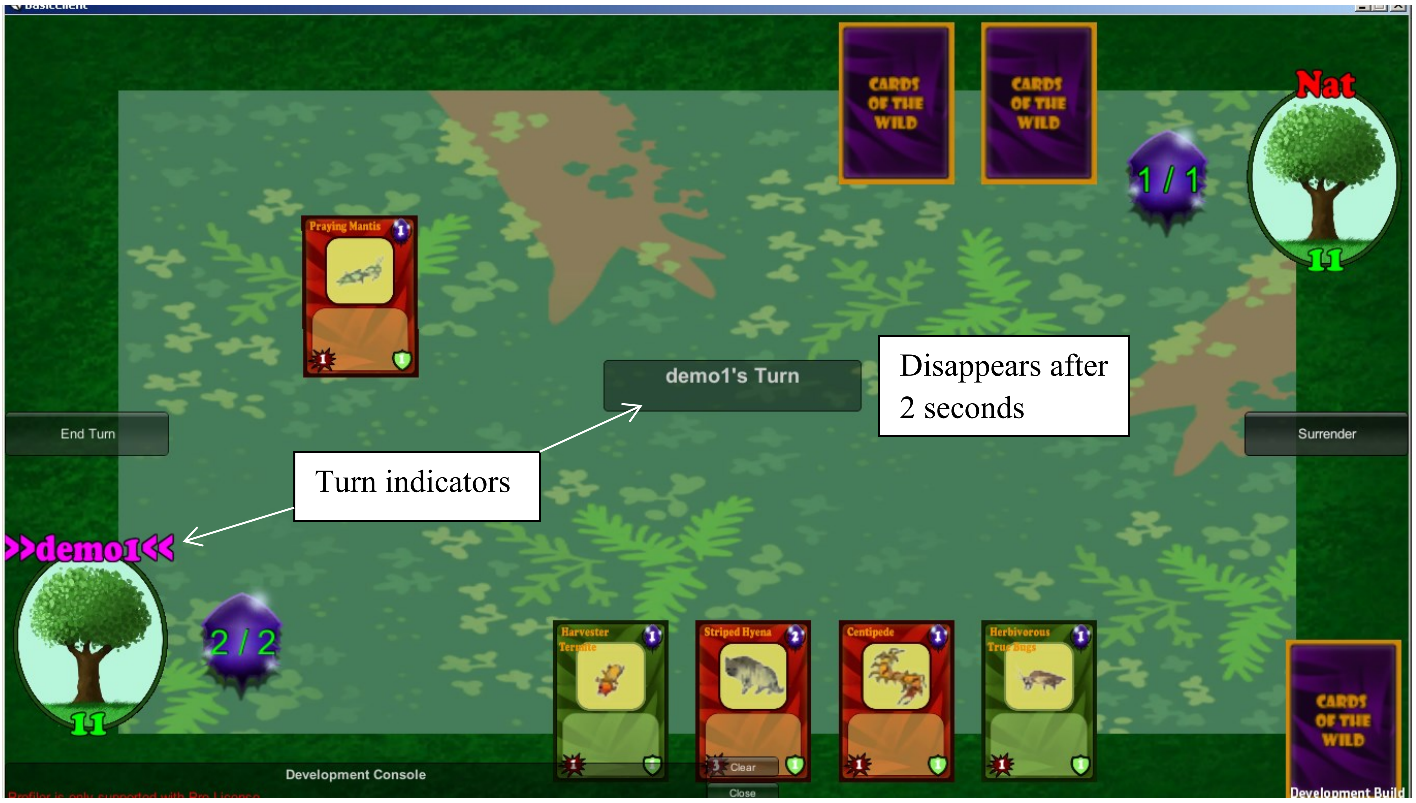
Dying

Turn Indicator



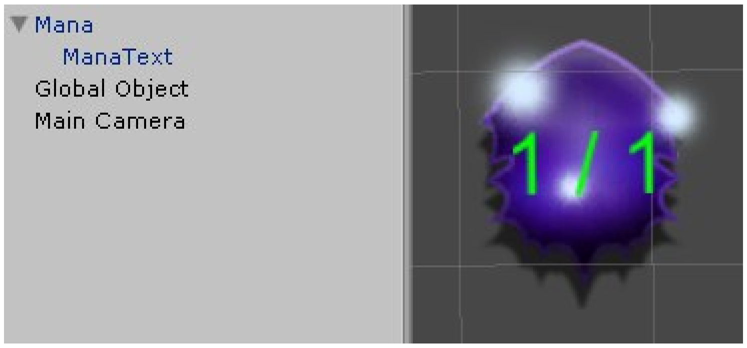
```
    void OnGUI(){
        if (showTurn > 0 && isActive) { //Shows active player
            GUI.skin.box.fontStyle = FontStyle.Bold;
            GUI.skin.box.fontSize = 20;
            GUI.Box(new Rect((Screen.width/ 2.0f)-100, (Screen.height/ 2.0f)-50, 250, 50), playerName+"'s Turn");
        }
        if (isGameOver) {
            if(GUI.Button(new Rect((Screen.width/2.0f)-((Screen.width/12.8f)/100 *150)/2.0f, //left
                                   ((Screen.height*2.0f)/3.0f), //height
                                   (Screen.width/12.8f)/100 *150,
                                   (Screen.width/12.8f)/100 *40),
                          "Back to Lobby")){
                GameManager.protocols.sendReturnToLobby();
            }
        }
```

In Game

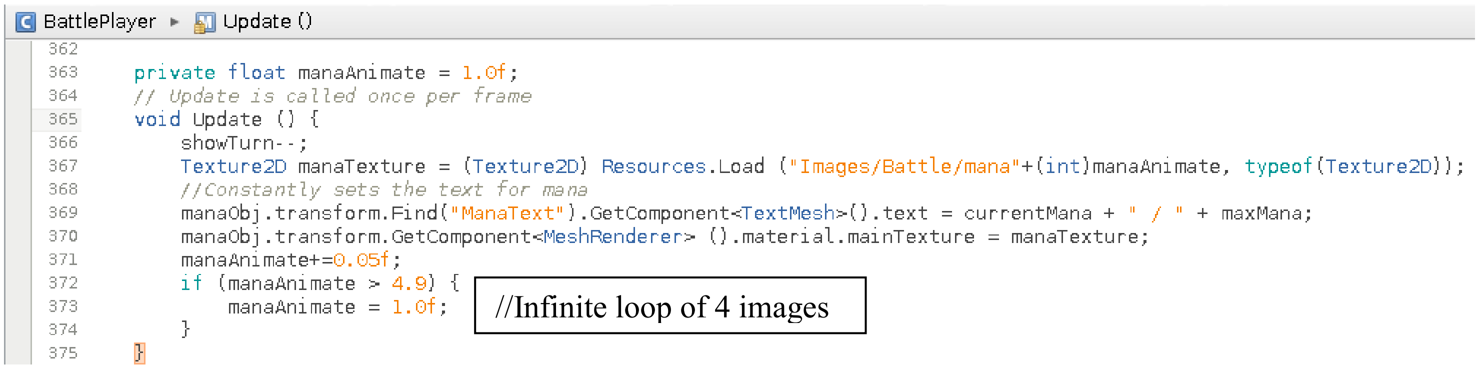


For the Resources/Mana, since Sonja provided 4 images of a mana crystal, I had the game Client cycle through each one in the Update() method, giving it an animated image. In addition, the ManaText would update how much maximum resources the player had and how much left the player could use as a fraction.

Mana Prefab

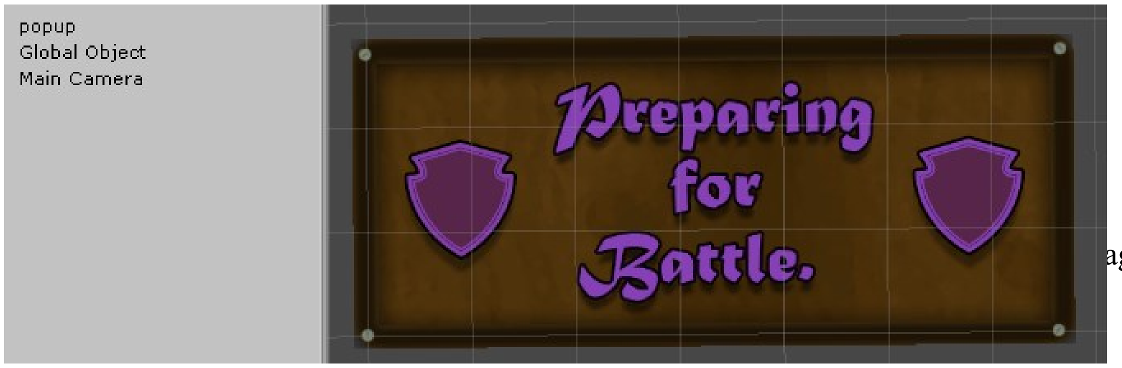


Mana Animate Code

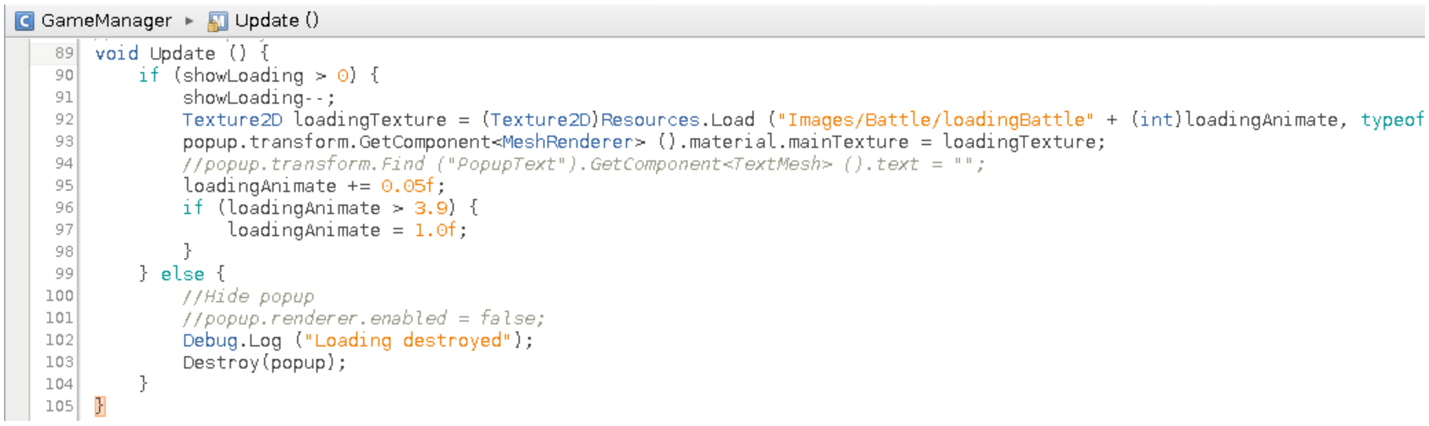


```
private float manaAnimate = 1.0f;
// Update is called once per frame
void Update () {
    showTurn--;
    Texture2D manaTexture = (Texture2D) Resources.Load ("Images/Battle/mana"+(int)manaAnimate, typeof(Texture2D));
    //Constantly sets the text for mana
    manaObj.transform.Find("ManaText").GetComponent<TextMesh>().text = currentMana + " / " + maxMana;
    manaObj.transform.GetComponent<MeshRenderer> ().material.mainTexture = manaTexture;
    manaAnimate+=0.05f;
    if (manaAnimate > 4.9) {
        manaAnimate = 1.0f;    //Infinite loop of 4 images
    }
}
```

I did a similar thing to the loading screen, though with this instance, I destroyed the gameObject after 5 seconds.

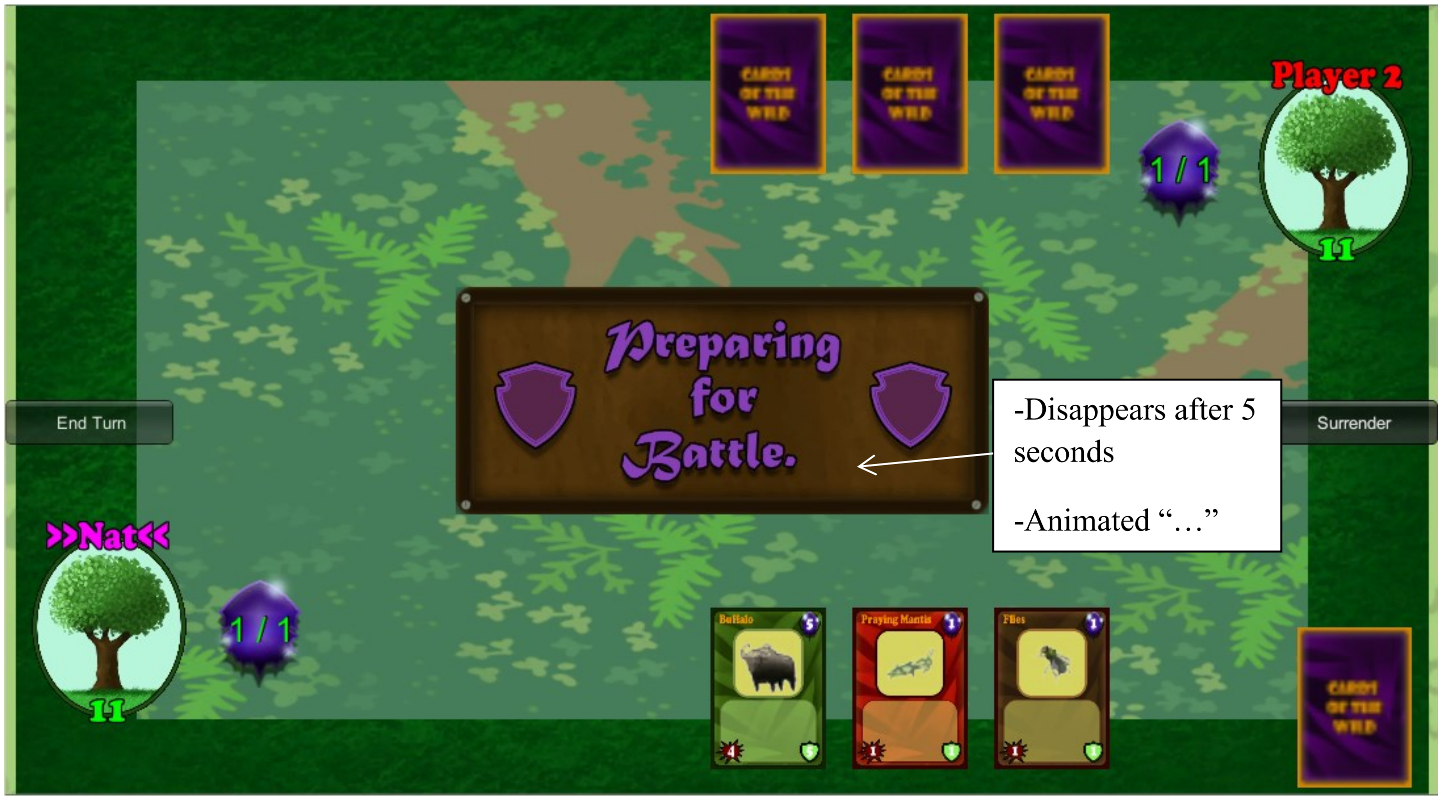


Loading Object



```
void Update () {
    if (showLoading > 0) {
        showLoading--;
        Texture2D loadingTexture = (Texture2D)Resources.Load ("Images/Battle/loadingBattle" + (int)loadingAnimate, typeof
        popup.transform.GetComponent<MeshRenderer> ().material.mainTexture = loadingTexture;
        //popup.transform.Find ("PopupText").GetComponent<TextMesh> ().text = "";
        loadingAnimate += 0.05f;
        if (loadingAnimate > 3.9) {
            loadingAnimate = 1.0f;
        }
    } else {
        //Hide popup
        //popup.renderer.enabled = false;
        Debug.Log ("Loading destroyed");
        Destroy(popup);
    }
}
```

Loading in-game

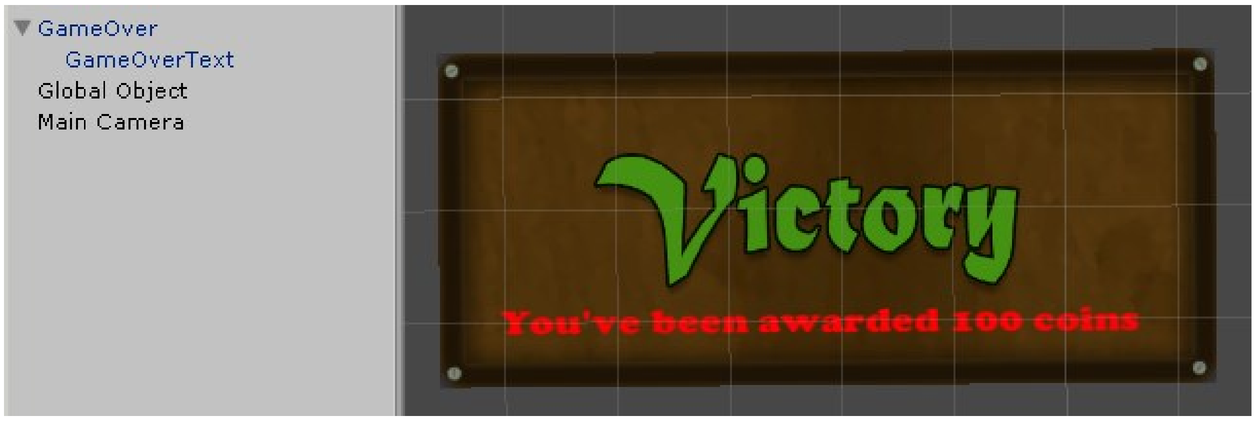


-Disappears after 5 seconds

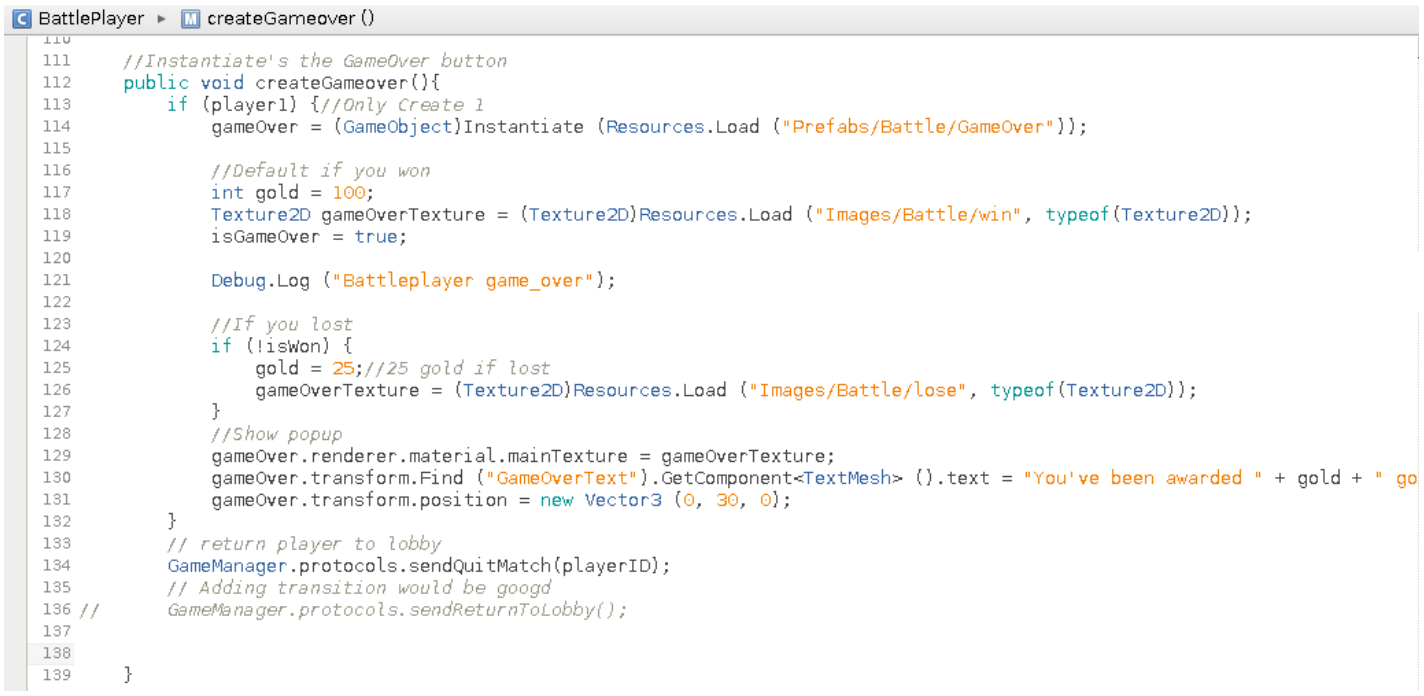
-Animated "…"

Finally, to end the game, there would need to be a button to return to the lobby. Therefore something needed to be loaded once the game was over. Because the Lobby wanted to give the player some sort of currency for a win or loss, I created a Boolean called isWon that defaulted to true, but became false when there were no more cards in their deck, their tree had <=0, or they surrendered. For surrendering, since I didn't want to jump directly from the game back to the lobby, I had a Surrender button that asked you what you wanted to do instead, so that you had a choice to continue the game.

GameOver Display Prefab

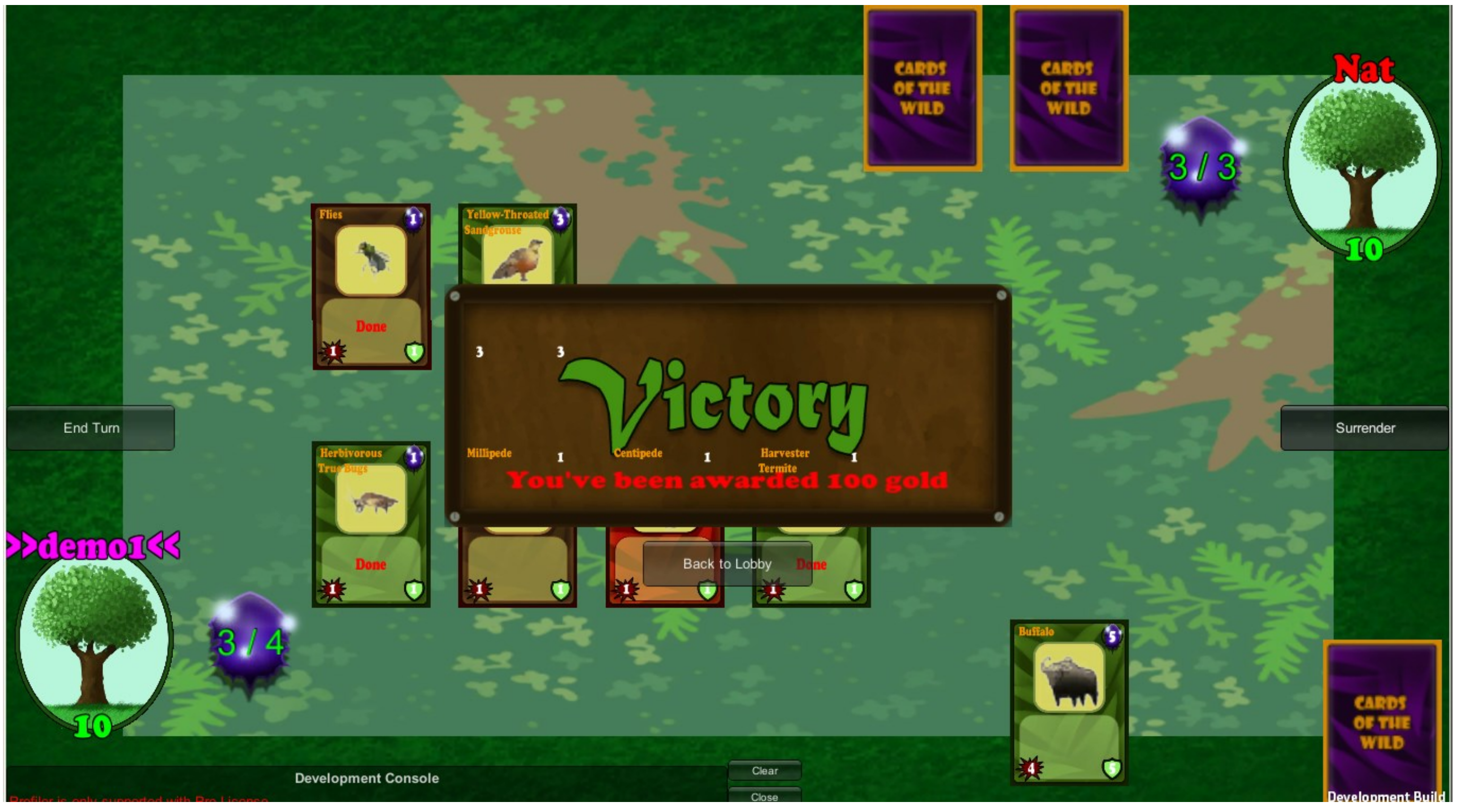


Changing prefab if won/lost

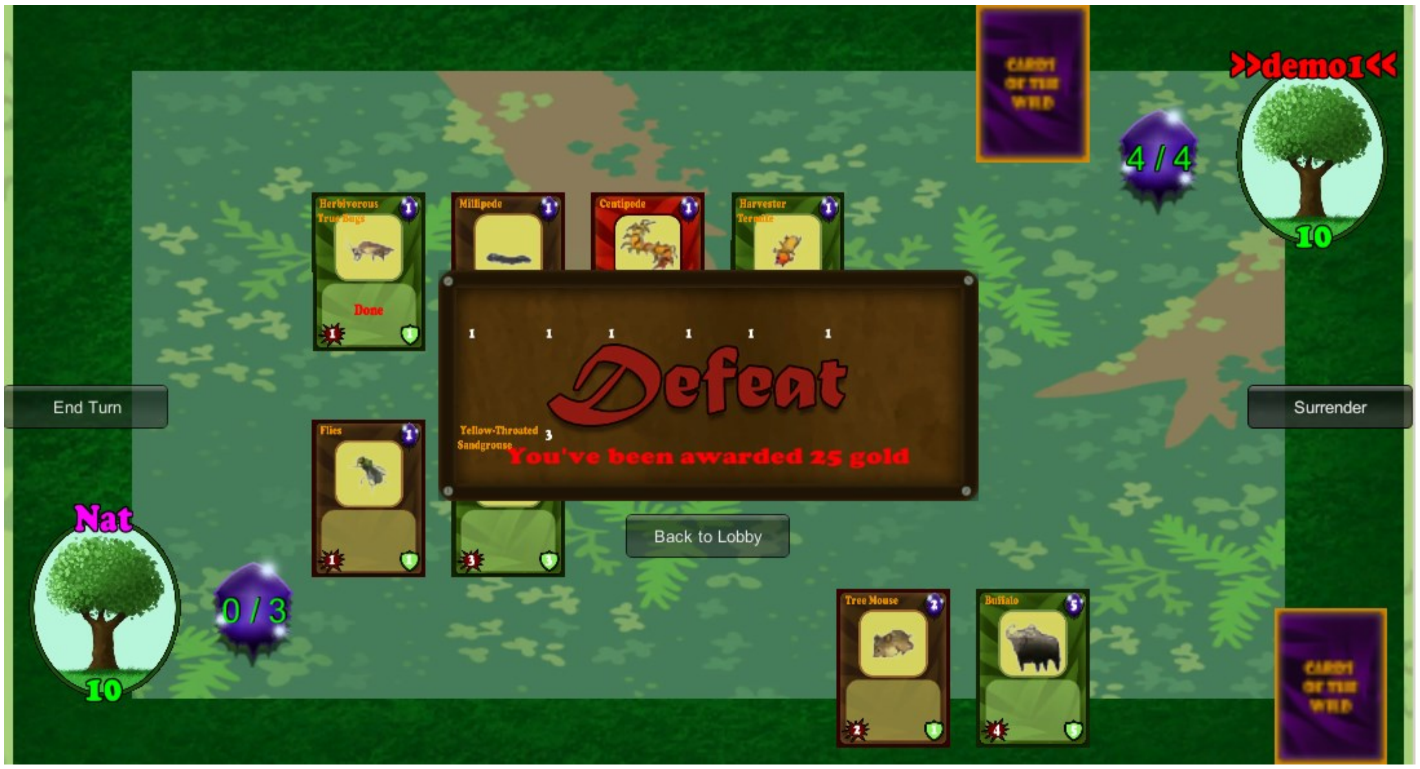


```
    //Instantiate's the GameOver button
    public void createGameover(){
        if (player1) {//Only Create 1
            gameOver = (GameObject)Instantiate (Resources.Load ("Prefabs/Battle/GameOver"));

            //Default if you won
            int gold = 100;
            Texture2D gameOverTexture = (Texture2D)Resources.Load ("Images/Battle/win", typeof(Texture2D));
            isGameOver = true;

            Debug.Log ("Battleplayer game_over");

            //If you lost
            if (!isWon) {
                gold = 25;//25 gold if lost
                gameOverTexture = (Texture2D)Resources.Load ("Images/Battle/lose", typeof(Texture2D));
            }
            //Show popup
            gameOver.renderer.material.mainTexture = gameOverTexture;
            gameOver.transform.Find ("GameOverText").GetComponent<TextMesh> ().text = "You've been awarded " + gold + " go
            gameOver.transform.position = new Vector3 (0, 30, 0);
        }
        // return player to lobby
        GameManager.protocols.sendQuitMatch(playerID);
        // Adding transition would be googd
//      GameManager.protocols.sendReturnToLobby();


    }
```

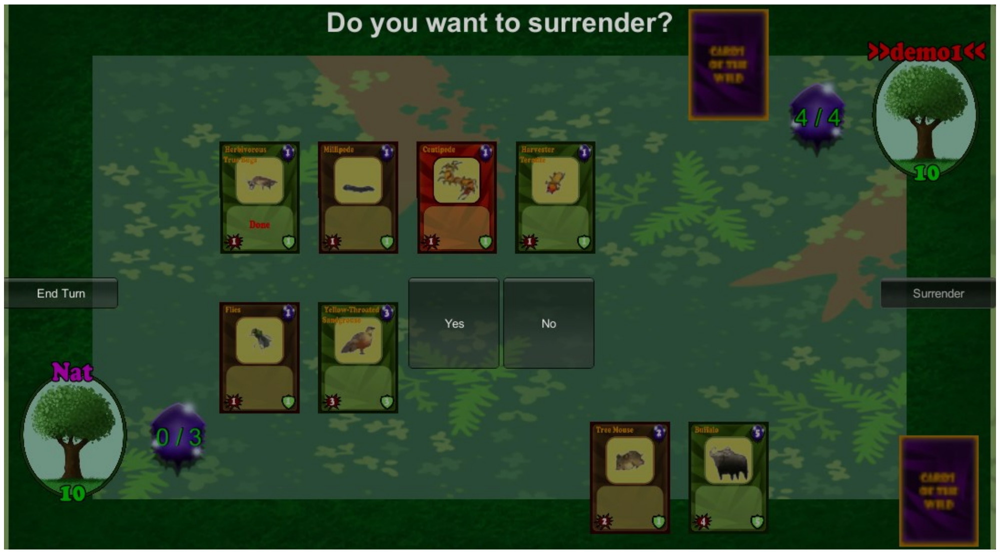
Victory

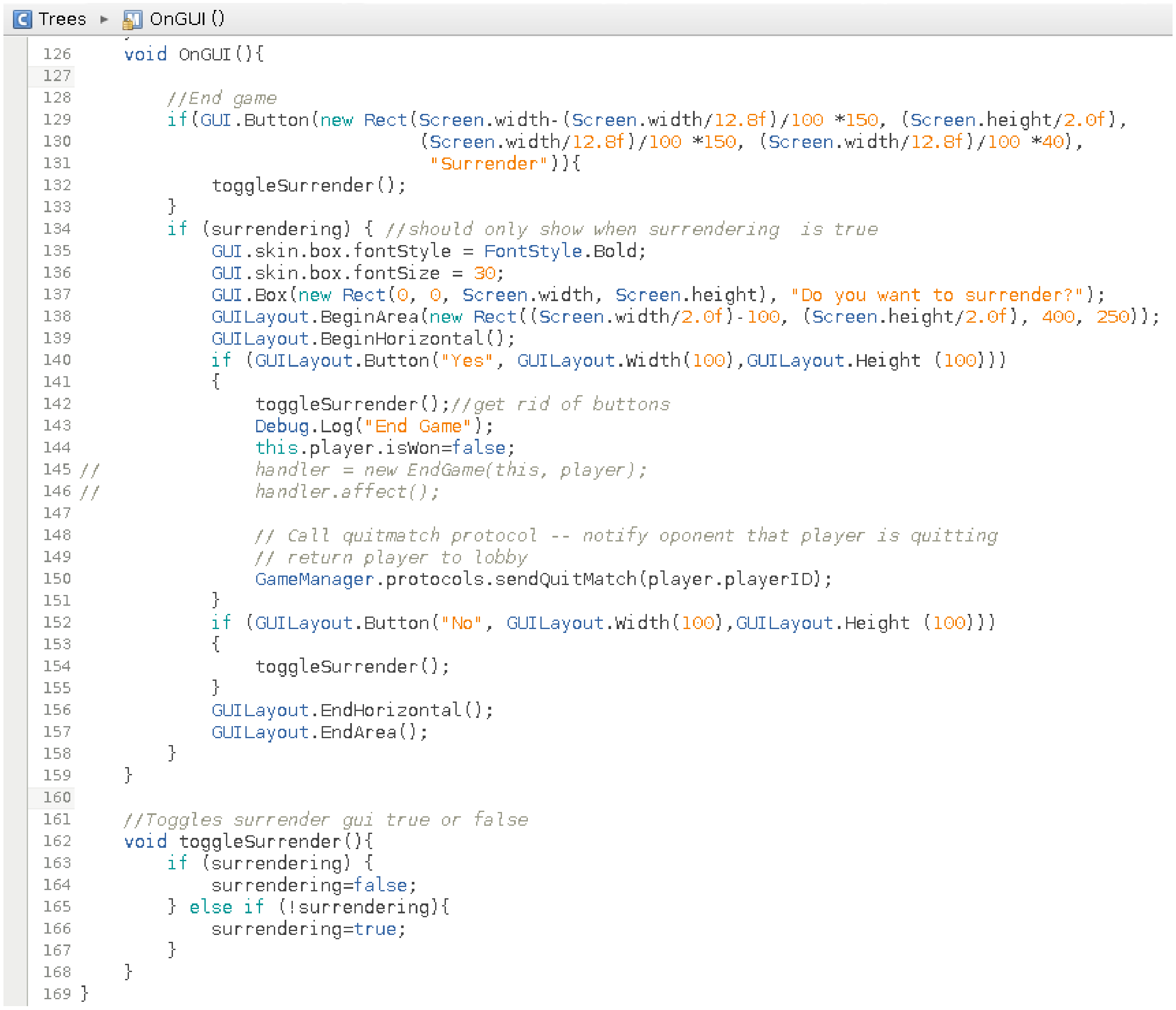


Defeat



Surrender Screen and code





```
    void OnGUI(){

        //End game
        if(GUI.Button(new Rect(Screen.width-(Screen.width/12.8f)/100 *150, (Screen.height/2.0f),
                                (Screen.width/12.8f)/100 *150, (Screen.width/12.8f)/100 *40),
                                 "Surrender")){
            toggleSurrender();
        }
        if (surrendering) { //should only show when surrendering  is true
            GUI.skin.box.fontStyle = FontStyle.Bold;
            GUI.skin.box.fontSize = 30;
            GUI.Box(new Rect(0, 0, Screen.width, Screen.height), "Do you want to surrender?");
            GUILayout.BeginArea(new Rect((Screen.width/2.0f)-100, (Screen.height/2.0f), 400, 250));
            GUILayout.BeginHorizontal();
            if (GUILayout.Button("Yes", GUILayout.Width(100),GUILayout.Height (100)))
            {
                toggleSurrender();//get rid of buttons
                Debug.Log("End Game");
                this.player.isWon=false;
//              handler = new EndGame(this, player);
//              handler.affect();

                // Call quitmatch protocol -- notify oponent that player is quitting
                // return player to lobby
                GameManager.protocols.sendQuitMatch(player.playerID);
            }
            if (GUILayout.Button("No", GUILayout.Width(100),GUILayout.Height (100)))
            {
                toggleSurrender();
            }
            GUILayout.EndHorizontal();
            GUILayout.EndArea();
        }
    }

    //Toggles surrender gui true or false
    void toggleSurrender(){
        if (surrendering) {
            surrendering=false;
        } else if (!surrendering){
            surrendering=true;
        }
    }
}
```

I also helped in documenting information on the game, including making slides for the Progress Report on April 7th, the Milestone Documentation Presentation on April 22nd, and the Final Documentation Presentation on May 2nd.

Documentation

- **Progress report 4/7:** https://docs.google.com/presentation/d/1s4ycTzzQb6w3JqKT4GjyurLaLFP2BFIqFn5Y4qcJuys
- **Milestone Documentation Presentation:** https://docs.google.com/presentation/d/1eCrBHGm_6j7HW8XdzXMXwaElUTF4vL52TiUmRNXHv24
- **Final Documentation Presentation:** https://docs.google.com/presentation/d/1W9cgkk4ikgpTFjTECJdX8ipuQvbnxyLZrX7ra6u6svQ

Other Links

- **Game Mechanics Comparison:** https://docs.google.com/spreadsheets/d/1E6Zq7Qx9re7k_NAi4Ic-B4wiOPWG9SXgxryT7ZKilq4
- **Species Data:** https://docs.google.com/spreadsheets/d/1cmBB1_og_EMvKyPUUE695P29-7ntMLYwmmbmXAcHD_0
- **Hearthstone Clone:** http://forum.ragezone.com/f857/unity-3d-hearthstone-clone-1034061/
- **Unity Trading Card Game Maker:** http://forum.unity3d.com/threads/trading-card-game-maker.237379/
- **Alpha Selective Shader:** http://forum.unity3d.com/threads/cant-make-another-color-transparent.213407/
- **One-Sided text Shader:** http://wiki.unity3d.com/index.php?title=3DText
- **Text-Outline script**: http://answers.unity3d.com/questions/542646/3d-text-strokeoutline.html