

# Touhou Card Game Documentation

Kailang Fu

Ellen Ni

Kuo Bao

Bingxu Han

Charlie Yang

Zhenglun Chen

Professor Marinov

CS 429

28 April 2019

# Table of Contents

Table of Contents	i
Project Description	1
End User Manual	1
Homepage Functionalities	1
Register/Log-in	1
Change Name and Bio	2
View Account Statistics	2
View In-Game Store	2
View Registered User	2
Create a Room	3
Join a Room	3
Start a Game	3
Game Manual	4
Draw One Card	4
End Turn	5
Summon a Monster	5
Place Spell Card	6
Apply Environment Card	7
Invoke Spell Card	8
Tribute Summon	8
Installation and Deployment	9
Process	9
Requirements & Specifications	10
Summary of User Stories	10
Architecture & Design	11
Backend	12
Frontend	13
Reflections and Lessons Learned	14
Kailang Fu	14
Ellen Ni	14

Kuo Bao	14
Bingxu Han	14
Charlie Yang	14
Zhenglun Chen	15
Appendix	16
Classes	16
Field	16
field.monsterSlots : Array.<MonsterCard>	16
field.spellSlots : Array.<SpellCard>	17
field.environmentSlot : EnvironmentCard	17
field.graveyard : Array.<Card>	17
field.oblivion : Array.<Card>	17
field.slotIds : Array.<String>	17
field.hasMonsterSlot(id) : Boolean	17
field.hasSpellSlot(id) : Boolean	17
field.isSlotEmpty(id) : Boolean	17
field.setSlot(id, card) : Card	17
field.getSlot(id) : Card	18
field.getMonsterSlotId(slotIndex) : String	18
field.getSpellSlotId(slotIndex) : String	18
field.removeCardFromOblivion(index) : Card	18
field.endTurn()	18
field.hasMonster() : Boolean	18
field.findSpellById(id) : SpellCard   null	18
field.findMonsterById(id) : MonsterCard   null	19
field.findCardById(id) : Card	19
field.removeCardById(id) : Object	19
field.killMonsterById(id)	19
field.killSpellById(id)	19
Game	19
new Game(users)	20

game.players : Array.<Player>	20
game.round : Number	20
game.turn : Number	20
game.playerIndexById : Object.<string, number>	20
game.isMyTurn(userId) : Boolean	20
game.invokeSpell(spellId, invokeParams) : any	21
game.invokeMonsterEffect(monsterId, invokeParams) : any	21
game.draw() : Object	21
game.summon(monsterId, slotId, display, pose, tributes) : Object	21
game.place(spellId, slotId, display) : Object	21
game.applyEnvironment(envId) : Object   Object	22
game.changeDisplay(monsterId, display) : Object	22
game.changePose(monsterId, pose) : Object	22
game.attack(monsterId, targetMonsterId) : Object	22
game.directAttack(monsterId, targetPlayerId) : Object	22
game.endTurn() : Object	22
game.checkGameEnd() : boolean	23
game.takeSnapshot() : Object	23
game.findCardOwnerById(cardId) : Player   null	23
game.findPlayer(userId) : Player   null	23
Game.success() : Object	23
Game.error(msg) : Object	23
Player	23
new Player(user)	24
player.deck : Array.<Card>	24
player.hand : Array.<Card>	24
player.hasActivated : Object.<string, boolean>	24
player.canDraw() : Boolean	24
player.draw()	24
player.endTurn()	24
player.findCardInDeckByName(name) : Number	25

player.removeCardFromDeck(index) : Card	25
player.canBeDirectlyAttacked() : Boolean	25
player.directAttack(monster, targetUser)	25
player.attack(monster, targetUser, targetMonster)	25
player.receiveDamage(attack)	25
player.removeCardInHand(index) : Card	25
player.findCardInHandById(id) : Card   null	26
player.removeCardInHandById(id)	26
Classes	26
BlueEyesWhiteDragonCard - MonsterCard	27
blueEyesWhiteDragonCard.canSummon(display, pose) : Boolean	27
blueEyesWhiteDragonCard.summon(display, pose)	28
blueEyesWhiteDragonCard.canChangeDisplay(display) : Boolean	28
blueEyesWhiteDragonCard.canChangePose(pose) : Boolean	28
blueEyesWhiteDragonCard.changeDisplay(display)	28
blueEyesWhiteDragonCard.changePose(pose)	28
blueEyesWhiteDragonCard.canAttack() : Boolean	28
blueEyesWhiteDragonCard.attack()	29
blueEyesWhiteDragonCard.endTurn()	29
blueEyesWhiteDragonCard.takeSnapshot() : Object	29
blueEyesWhiteDragonCard.canInvoke(game, player, invokeParams) : Boolean	29
blueEyesWhiteDragonCard.invoke(game, player, invokeParams)	29
blueEyesWhiteDragonCard.canPlace(display) : Boolean	29
blueEyesWhiteDragonCard.place(display)	29
Card	30
new Card(name, desc, imgUrl)	30
card.endTurn()	30
card.canInvoke(game, player, invokeParams) : Boolean	30
card.invoke(game, player, invokeParams)	30
card.canSummon(display, pose) : Boolean	31

card.summon(display, pose)	31
card.canPlace(display) : Boolean	31
card.place(display)	31
card.takeSnapshot() : Object	31
Card.generateId(len) : String	31
Card.createAction(name, desc, position, params) : Object	32
Card.createActionParam(select, position, owner, desc) : Object	32
DarkMagicAttackCard - SpellCard	32
darkMagicAttackCard.canInvoke(game, player) : Boolean	32
darkMagicAttackCard.invoke(game, player, invokeParams)	33
darkMagicAttackCard.takeSnapshot() : Object	33
darkMagicAttackCard.canPlace() : Boolean	33
darkMagicAttackCard.place(display)	33
DarkMagicianCard - MonsterCard	33
darkMagicianCard.canSummon(display, pose) : Boolean	34
darkMagicianCard.summon(display, pose)	34
darkMagicianCard.canChangeDisplay(display) : Boolean	34
darkMagicianCard.canChangePose(pose) : Boolean	34
darkMagicianCard.changeDisplay(display)	34
darkMagicianCard.changePose(pose)	34
darkMagicianCard.canAttack() : Boolean	35
darkMagicianCard.attack()	35
darkMagicianCard.endTurn()	35
darkMagicianCard.takeSnapshot() : Object	35
darkMagicianCard.canInvoke(game, player, invokeParams) : Boolean	35
darkMagicianCard.invoke(game, player, invokeParams)	35
darkMagicianCard.canPlace(display) : Boolean	35
darkMagicianCard.place(display)	36
DarkMagicianGirlCard - MonsterCard	36
darkMagicianGirlCard.canInvoke() : Boolean	36
darkMagicianGirlCard.invoke(game, player)	36

darkMagicianGirlCard.takeSnapshot() : Object	37
darkMagicianGirlCard.canSummon(display, pose) : Boolean	37
darkMagicianGirlCard.summon(display, pose)	37
darkMagicianGirlCard.canChangeDisplay(display) : Boolean	37
darkMagicianGirlCard.canChangePose(pose) : Boolean	37
darkMagicianGirlCard.changeDisplay(display)	37
darkMagicianGirlCard.changePose(pose)	37
darkMagicianGirlCard.canAttack() : Boolean	38
darkMagicianGirlCard.attack()	38
darkMagicianGirlCard.endTurn()	38
darkMagicianGirlCard.canPlace(display) : Boolean	38
darkMagicianGirlCard.place(display)	38
DarkMagicVeilCard - SpellCard	38
darkMagicVeilCard.canInvoke(game, player, invokeParams) : Boolean	39
darkMagicVeilCard.invoke(game, player, invokeParams)	39
darkMagicVeilCard.takeSnapshot() : Object	39
darkMagicVeilCard.canPlace() : Boolean	39
darkMagicVeilCard.place(display)	39
EnvironmentCard - Card	39
new EnvironmentCard(name, desc, imgUrl)	40
environmentCard.place(display)	40
environmentCard.takeSnapshot() : Object	40
environmentCard.endTurn()	40
environmentCard.canInvoke(game, player, invokeParams) : Boolean	40
environmentCard.invoke(game, player, invokeParams)	40
environmentCard.canSummon(display, pose) : Boolean	41
environmentCard.summon(display, pose)	41
environmentCard.canPlace(display) : Boolean	41
FirestormMonarchCard - MonsterCard	41
firestormMonarchCard.canInvoke() : Boolean	42
firestormMonarchCard.invoke(game, player, invokeParams)	42

firestormMonarchCard.takeSnapshot() : Object	42
firestormMonarchCard.canSummon(display, pose) : Boolean	42
firestormMonarchCard.summon(display, pose)	42
firestormMonarchCard.canChangeDisplay(display) : Boolean	43
firestormMonarchCard.canChangePose(pose) : Boolean	43
firestormMonarchCard.changeDisplay(display)	43
firestormMonarchCard.changePose(pose)	43
firestormMonarchCard.canAttack() : Boolean	43
firestormMonarchCard.attack()	43
firestormMonarchCard.endTurn()	43
firestormMonarchCard.canPlace(display) : Boolean	43
firestormMonarchCard.place(display)	44
KaibamanCard - MonsterCard	44
kaibamanCard.canInvoke(game, player) : Boolean	44
kaibamanCard.invoke(game, player)	44
kaibamanCard.takeSnapshot() : Object	45
kaibamanCard.canSummon(display, pose) : Boolean	45
kaibamanCard.summon(display, pose)	45
kaibamanCard.canChangeDisplay(display) : Boolean	45
kaibamanCard.canChangePose(pose) : Boolean	45
kaibamanCard.changeDisplay(display)	45
kaibamanCard.changePose(pose)	46
kaibamanCard.canAttack() : Boolean	46
kaibamanCard.attack()	46
kaibamanCard.endTurn()	46
kaibamanCard.canPlace(display) : Boolean	46
kaibamanCard.place(display)	46
MobiusTheFrostMonarchCard - MonsterCard	46
mobiusTheFrostMonarchCard.canInvoke(game, player, invokeParams) :	
boolean	47
mobiusTheFrostMonarchCard.invoke(game, player, invokeParams)	47



mobiusTheFrostMonarchCard.takeSnapshot() : Object	47
mobiusTheFrostMonarchCard.canSummon(display, pose) : Boolean	48
mobiusTheFrostMonarchCard.summon(display, pose)	48
mobiusTheFrostMonarchCard.canChangeDisplay(display) : Boolean	48
mobiusTheFrostMonarchCard.canChangePose(pose) : Boolean	48
mobiusTheFrostMonarchCard.changeDisplay(display)	48
mobiusTheFrostMonarchCard.changePose(pose)	48
mobiusTheFrostMonarchCard.canAttack() : Boolean	48
mobiusTheFrostMonarchCard.attack()	49
mobiusTheFrostMonarchCard.endTurn()	49
mobiusTheFrostMonarchCard.canPlace(display) : Boolean	49
mobiusTheFrostMonarchCard.place(display)	49
MonsterCard - Card	49
new MonsterCard(name, desc, imgUrl, lv, atk, dfs)	50
monsterCard.canSummon(display, pose) : Boolean	50
monsterCard.summon(display, pose)	50
monsterCard.canChangeDisplay(display) : Boolean	50
monsterCard.canChangePose(pose) : Boolean	50
monsterCard.changeDisplay(display)	51
monsterCard.changePose(pose)	51
monsterCard.canAttack() : Boolean	51
monsterCard.attack()	51
monsterCard.endTurn()	51
monsterCard.takeSnapshot() : Object	51
monsterCard.canInvoke(game, player, invokeParams) : Boolean	51
monsterCard.invoke(game, player, invokeParams)	51
monsterCard.canPlace(display) : Boolean	52
monsterCard.place(display)	52
PotOfGreedCard - SpellCard	52
potOfGreedCard.invoke(game, player)	52
potOfGreedCard.takeSnapshot() : Object	52

potOfGreedCard.canPlace() : Boolean	53
potOfGreedCard.place(display)	53
potOfGreedCard.canInvoke(game, player, invokeParams) : Boolean	53
RaizaTheStormMonarchCard - MonsterCard	53
raizaTheStormMonarchCard.invoke(game, player, invokeParams)	54
raizaTheStormMonarchCard.takeSnapshot() : Object	54
raizaTheStormMonarchCard.canSummon(display, pose) : Boolean	54
raizaTheStormMonarchCard.summon(display, pose)	54
raizaTheStormMonarchCard.canChangeDisplay(display) : Boolean	54
raizaTheStormMonarchCard.canChangePose(pose) : Boolean	54
raizaTheStormMonarchCard.changeDisplay(display)	55
raizaTheStormMonarchCard.changePose(pose)	55
raizaTheStormMonarchCard.canAttack() : Boolean	55
raizaTheStormMonarchCard.attack()	55
raizaTheStormMonarchCard.endTurn()	55
raizaTheStormMonarchCard.canInvoke(game, player, invokeParams) : Boolean	55
raizaTheStormMonarchCard.canPlace(display) : Boolean	55
raizaTheStormMonarchCard.place(display)	56
SageStoneCard - SpellCard	56
sageStoneCard.canInvoke(game, player, invokeParams) : Boolean	56
sageStoneCard.invoke(game, player, invokeParams)	56
sageStoneCard.takeSnapshot() : Object	56
sageStoneCard.canPlace() : Boolean	57
sageStoneCard.place(display)	57
SorcerousSpellWallCard - EnvironmentCard	57
sorcerousSpellWallCard.applyEnvironment(monsterCard)	57
sorcerousSpellWallCard.place(display)	57
sorcerousSpellWallCard.takeSnapshot() : Object	58
sorcerousSpellWallCard.endTurn()	58
sorcerousSpellWallCard.canInvoke(game, player, invokeParams) : Boolean	58

sorcerousSpellWallCard.invoke(game, player, invokeParams)	58
sorcerousSpellWallCard.canSummon(display, pose) : Boolean	58
sorcerousSpellWallCard.summon(display, pose)	58
sorcerousSpellWallCard.canPlace(display) : Boolean	58
SpellbookOfEternityCard - SpellCard	59
spellbookOfEternityCard.canPlace() : Boolean	59
spellbookOfEternityCard.place(display)	59
spellbookOfEternityCard.canInvoke(game, player, invokeParams) : Boolean	59
spellbookOfEternityCard.takeSnapshot() : Object	59
SpellbookOfSecretsCard - SpellCard	59
spellbookOfSecretsCard.canInvoke(game, player, invokeParams) : Boolean	60
spellbookOfSecretsCard.invoke(game, player, invokeParams)	60
spellbookOfSecretsCard.canPlace() : Boolean	60
spellbookOfSecretsCard.place(display)	60
spellbookOfSecretsCard.takeSnapshot() : Object	60
ThousandKnivesCard - SpellCard	61
thousandKnivesCard.canInvoke(game, player, invokeParams) : Boolean	61
thousandKnivesCard.invoke(game, player, invokeParams)	61
thousandKnivesCard.takeSnapshot() : Object	61
thousandKnivesCard.canPlace() : Boolean	61
thousandKnivesCard.place(display)	61
TwistedSpaceCard - EnvironmentCard	62
twistedSpaceCard.applyEnvironment(monsterCard)	62
twistedSpaceCard.place(display)	62
twistedSpaceCard.takeSnapshot() : Object	62
twistedSpaceCard.endTurn()	62
twistedSpaceCard.canInvoke(game, player, invokeParams) : Boolean	62
twistedSpaceCard.invoke(game, player, invokeParams)	63
twistedSpaceCard.canSummon(display, pose) : Boolean	63
twistedSpaceCard.summon(display, pose)	63

twistedSpaceCard.canPlace(display) : Boolean	63
ZaborgTheThunderMonarchCard - MonsterCard	63
zaborgTheThunderMonarchCard.canInvoke(game, player, invokeParams) : Boolean	64
zaborgTheThunderMonarchCard.invoke(game, player, invokeParams)	64
zaborgTheThunderMonarchCard.takeSnapshot() : Object	64
zaborgTheThunderMonarchCard.canSummon(display, pose) : Boolean	64
zaborgTheThunderMonarchCard.summon(display, pose)	65
zaborgTheThunderMonarchCard.canChangeDisplay(display) : Boolean	65
zaborgTheThunderMonarchCard.canChangePose(pose) : Boolean	65
zaborgTheThunderMonarchCard.changeDisplay(display)	65
zaborgTheThunderMonarchCard.changePose(pose)	65
zaborgTheThunderMonarchCard.canAttack() : Boolean	65
zaborgTheThunderMonarchCard.attack()	66
zaborgTheThunderMonarchCard.endTurn()	66
zaborgTheThunderMonarchCard.canPlace(display) : Boolean	66
zaborgTheThunderMonarchCard.place(display)	66
Network APIs	67
Dependencies	68
Backend	68
Frontend	68

# Project Description

Touhou Card Game (TCG) is a web-based multiplayer online card game platform which uses characters from Touhou Project. Competing players use cards in their hand and deck to improve their field and deal damage to their opponents. Taking damage reduces a player's Life Points until the last player remaining wins. After playing, users can use earned Spirit Points to purchase upgrades.

## End User Manual

We have separated the End User Manual for TCG to two parts. The first part is the Homepage Functionalities and the second part and the Game Manual. We also include a part where the installation and deployment process is discussed.

### Homepage Functionalities

There are several things you can do on homepage: change name and bio, view account statistics, view store, view registered players, create room, start a game.

### Register/Log-in

After launching Touhou Card Game, this is the Register/Login page for Touhou Card Game.

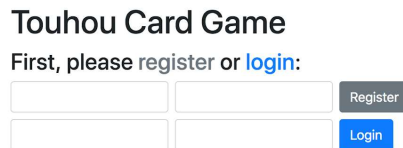


Figure 3.1. The Register/Login page for Touhou Card Game.

If you don't have account for Touhou Card Game, you can enter your intended username and password into the boxes on the first row and click Register button.

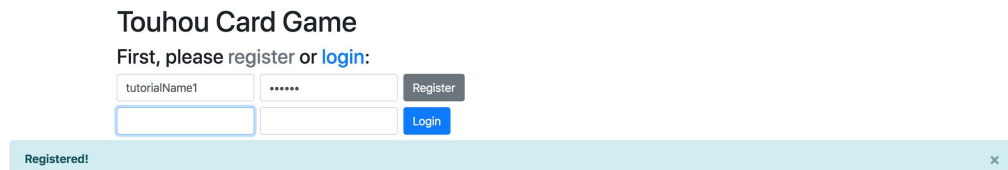


Figure 3.2. The successful message is shown.

After registering, or if you have an existing account, you can log in with your username and password by entering them into the boxes on the second row and click Login button. After logging-in, you will be directed to the homepage for Touhou Card Game.

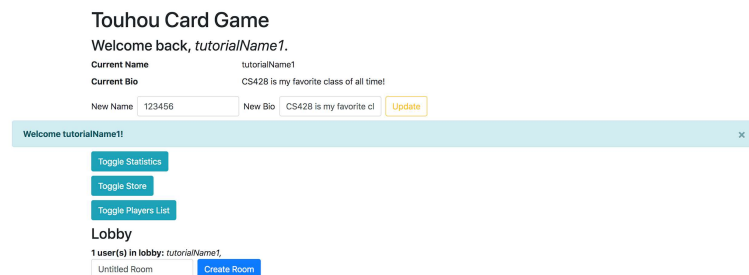


Figure 3.3. The homepage for TCG.

## Change Name and Bio

By entering your intended new name and bio in the responding boxes and click update, you can change your name and bio respectively. After changing name and bio, this page will show correspondingly.

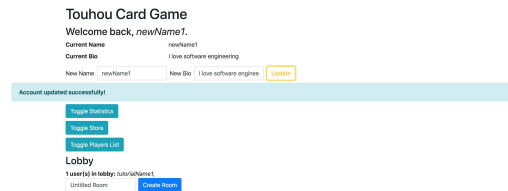


Figure 3.4. After changing name and bio.

## View Account Statistics

By clicking “Toggle Statistics” button, you can view your account statistics including the following records.

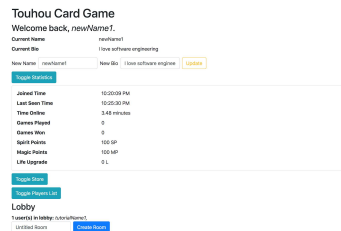


Figure 3.5. The player’s account statistics.

## View In-Game Store

By clicking “Toggle Store”, you can view the in-game store to buy extra 100LP with 50SP.

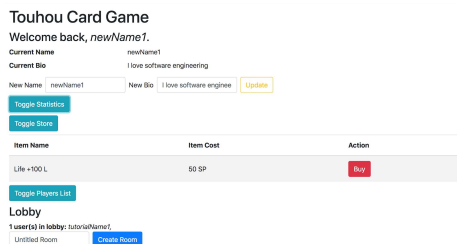


Figure 3.6. The in-game store.

## View Registered User

By clicking “Toggle Players List”, you can view the name and bio of all registered users.

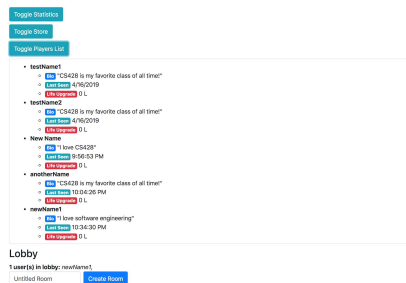


Figure 3.7. The player list.

## Create a Room

By clicking “Create Room”, you can create your room and other players can see it and join it.

The screenshot shows the 'Touhou Card Game' lobby. At the top, it says 'Welcome back, newName1.' Below this, there are fields for 'Current Name' (newName1) and 'Current Bio' (I love software engineering). There are also input fields for 'New Name' (123456) and 'New Bio' (I love software enginee) with an 'Update' button. Below these are three toggle buttons: 'Toggle Statistics', 'Toggle Store', and 'Toggle Players List'. The 'Lobby' section shows 'Joined room: my room (owned by newName1)' and a 'Leave Room' button. It also says 'Not enough members.' and 'Chat' with a text input and 'Send' button.

Figure 3.8. A room is created.

## Join a Room

Also, as a player, you can also choose to join any existing room by clicking “Join” button for the room you want to join.

The screenshot shows the 'Touhou Card Game' lobby. At the top, it says 'Welcome back, anotherName.' Below this, there are fields for 'Current Name' (anotherName) and 'Current Bio' (CS428 is my favorite class of all time!). There are also input fields for 'New Name' (123456) and 'New Bio' (CS428 is my favorite cl) with an 'Update' button. Below these are three toggle buttons: 'Toggle Statistics', 'Toggle Store', and 'Toggle Players List'. The 'Lobby' section shows '1 user(s) in lobby: anotherName,' and 'Unfilled Room' with a 'Create Room' button. Below this, it shows 'my room (owned by newName1)' with a 'Join' button.

Figure 3.9. The player can select a room to join.

## Start a Game

If you are the owner of the room, you can choose to propose to start a game clicking the button so that other players can choose to agree or refuse to start the game.

The screenshot shows the 'Touhou Card Game' lobby. At the top, it says 'Welcome back, New Name.' Below this, there are fields for 'Current Name' (New Name) and 'Current Bio' (I love CS428). There are also input fields for 'New Name' (123456) and 'New Bio' (I love CS428) with an 'Update' button. Below these are three toggle buttons: 'Toggle Statistics', 'Toggle Store', and 'Toggle Players List'. The 'Lobby' section shows 'Joined room: my room (owned by New Name) 1 members: anotherName' and a 'Leave Room' button. It also says 'You are the host' and 'Propose to Start a Game' button. Below this is a 'Chat' section with a text input and 'Send' button.

Figure 3.10. The owner of the room can start a game.

Other players in the room can agree to start the game or refuse to start the game.

The screenshot shows the 'Touhou Card Game' lobby. At the top, it says 'Welcome back, anotherName.' Below this, there are fields for 'Current Name' (anotherName) and 'Current Bio' (CS428 is my favorite class of all time!). There are also input fields for 'New Name' (123456) and 'New Bio' (CS428 is my favorite cl) with an 'Update' button. Below these are three toggle buttons: 'Toggle Statistics', 'Toggle Store', and 'Toggle Players List'. The 'Lobby' section shows 'Joined room: my room (owned by New Name) 1 members: anotherName' and a 'Leave Room' button. It also says 'anotherName Waiting for agreement' and 'Agree to Start the Game' and 'Refuse to Start the Game' buttons. Below this is a 'Chat' section with a text input and 'Send' button.

Figure 3.11. Other players' options.

If the other player choose to refuse to start the game, the player will automatically leave the room.

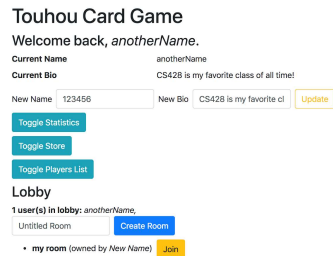


Figure 3.12. The player is forced to leave the room.

If more than two players choose to agree to start the game, the owner can click on “Start the Game” to launch the actual game.

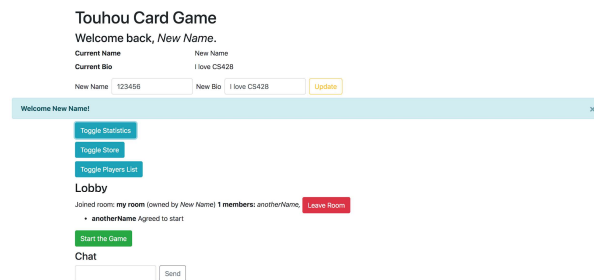


Figure 3.13. The room owner can start the game.

This is the view after starting the game.



Figure 3.14. The game is started.

## Game Manual

### Draw One Card

You can draw one card every turn.

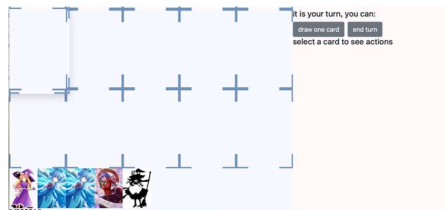


Figure 3.15. Before drawing the card.



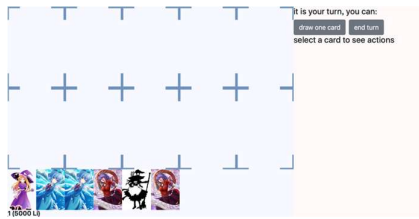


Figure 3.16. After drawing the card.

## End Turn

You can end your turn when you do not have any actions to perform.

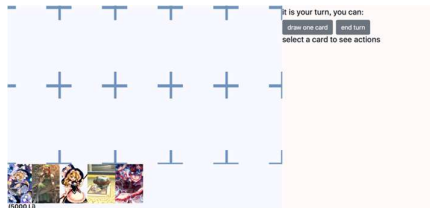


Figure 3.17. Before ending your turn.



Figure 3.18. After ending your turn.

## Summon a Monster

If you have a monster in your hand that can be summoned, you can summon the monster in your turn.

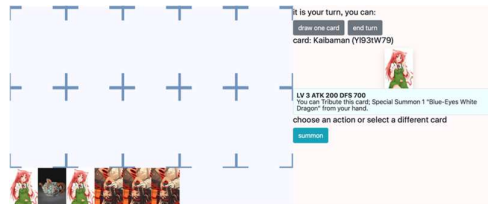


Figure 3.19. Select the card in your hand.

When you are summoning a monster, you have to choose the monster slot on your field where you want the monster to be summoned into.

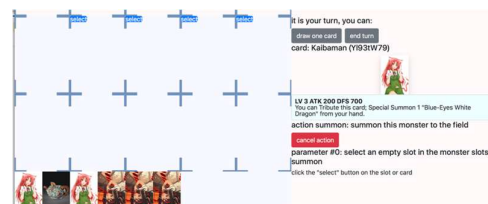


Figure 3.20. Select a monster slot.

Choose the monster's display from "Revealed" and "Hidden."

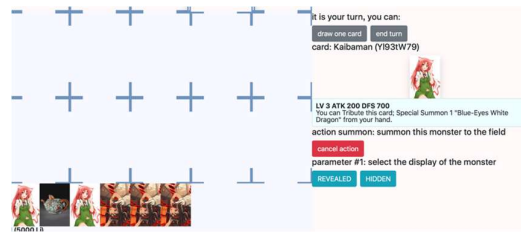


Figure 3.21. Choose the monster's display.

A monster can be in either “Attack” or “Defense” pose. You have to choose between “Attack” or “Defense” poses.

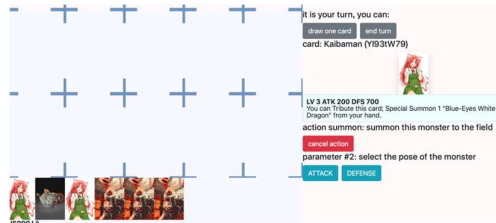


Figure 3.22. Choose between “Attack” or “Defense” poses.

Confirm your action to summon the monster.

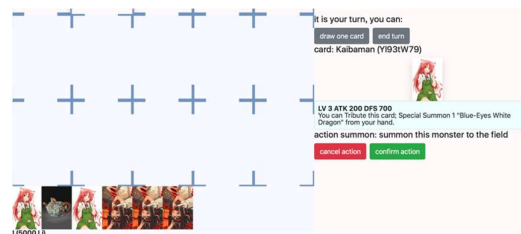


Figure 3.23. Choose to confirm the action.

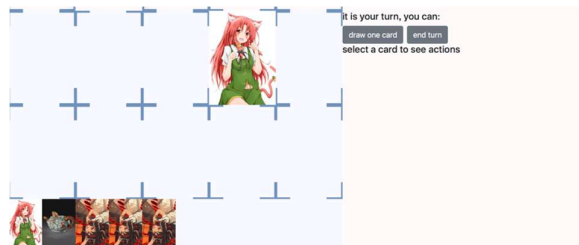


Figure 3.24. The monster is summoned.

## Place Spell Card

If you have a spell card, you can place it onto one of the spell slots on your field.

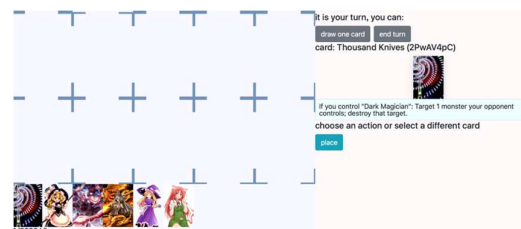


Figure 3.25. A spell card in your hand.

You have to select an empty spell slot to put the spell.

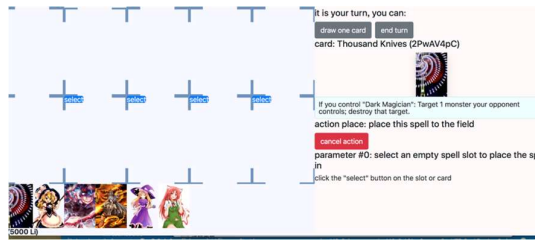


Figure 3.26. Selecting a spell slot.

Choose between "Revealed" or "Hidden."

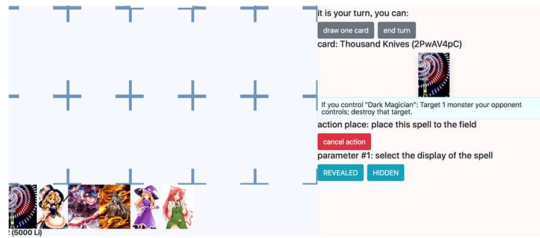


Figure 3.27. Choose between "Revealed" or "Hidden."

Confirm action.

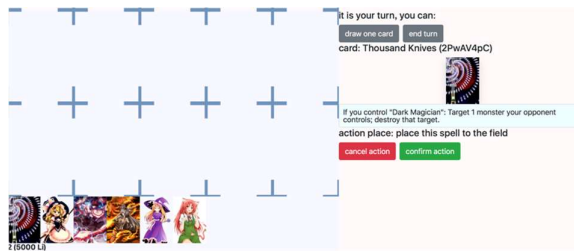


Figure 3.28. Confirm action.

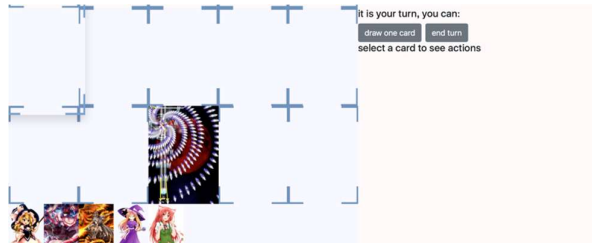


Figure 3.29. The spell card is placed.

## Apply Environment Card

You can apply an environment card to your field.

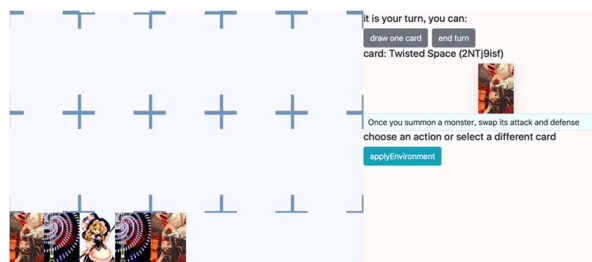


Figure 3.30. An environment card.

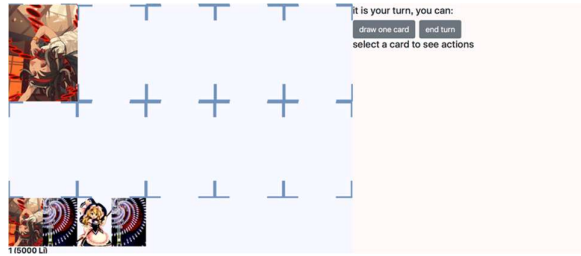


Figure 3.31. The environment card is applied.

## Invoke Spell Card

You can invoke a spell card on the field.

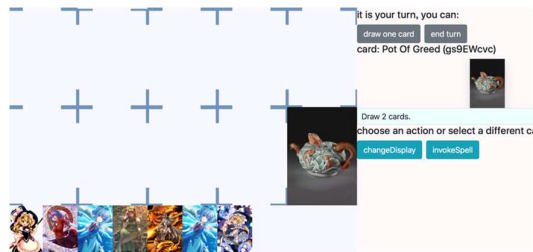


Figure 3.32. The actions for the spell card.

## Tribute Summon

You can summon a monster using one or two tributes.



Figure 3.33. The monster to be summoned using tributes.

Select the monster that you want to tribute.

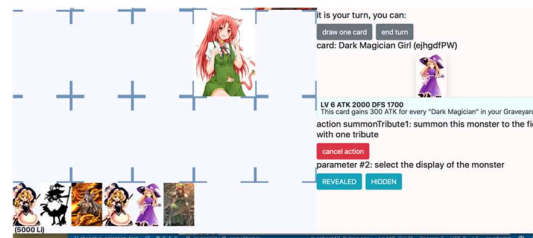


Figure 3.34. Select the monster that you want to tribute.

Choose the display and the pose of the monster.

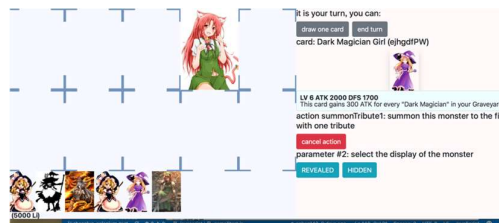


Figure 3.35. Select the display.

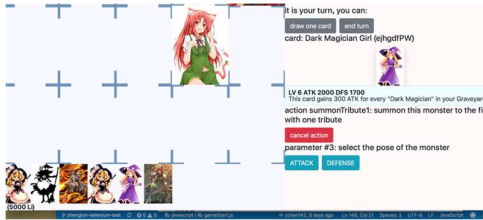


Figure 3.36. Select the pose.



Figure 3.37. The monster is summoned.

## Installation and Deployment

Install NodeJS 10.x LTS: <https://nodejs.org>

Install MongoDB: <https://www.mongodb.com/download-center/community>

Clone this repository:

```
git clone https://github.com/DotLab/touhou-card-game-nodejs
```

Go to the project root directory:

```
cd touhou-card-game-nodejs
```

Install back-end dependencies:

```
npm install
```

Build front end:

```
npm run build-app
```

Run production server on port 3000. This server also serves front end.

```
npm start
```

You can now access the Touhou Card Game without your browsers at <http://localhost:3000>. For production usage, you may want to use a web server like Nginx to setup a reverse proxy for the node backend in your server.

## Process

We used Extreme Programming methodology with some adjustments. Notably, we wrote tests after writing the main logic, and we only did pair programming when necessary. The adjustment to the Extreme Programming methodology is based on our status as college students. We do not have as much experience as industry professionals, nor do we work forty hours a week on one project.

First, we did not write tests before writing code. We do not expect that we will produce the right structure in the first iteration. Therefore, there is no benefit in writing the tests for faulty interfaces that cannot work as designed. Every time we find out that our design does not live up to their expectation, we would have to write another set of tests. Hence, for a team of college students, it is better to write the tests after finishing the real design so that we will have results to present and so that we will not waste our time in writing deleted tests.

Even though we did not follow Test Driven Development principles, we have a wide breadth and depth of tests. We have unit tests, integration tests, and advanced testing with Selenium, and our code coverage reached 100% line coverage.

Second, we did not enforce pair programming. As college students, our schedules can vary greatly from each other in contrast to coworkers who see each other for eight hours a day. In our team specifically, we had the least members. Pair programming would limit our code output too much. Although we did not require pair programming, we still collaborated with each other by consulting each other, which promoted a sense of mentorship. Additionally, we required all pull requests to be code reviewed by two other members. This allowed us to understand what other team members were doing, which is one of the benefits of pair programming.

Code review also allowed us to point out opportunities for refactoring to each other. In addition, we looked for chances to refactor whenever we wanted to use part of a feature we previously wrote. Each iteration we worked on both the game and the player interactions, because both were equally important to our project. For example, in one iteration we worked on a chat system, creating various cards, and placing cards onto the field.

## Requirements & Specifications

TCG is a platform that supports match making, following system, in-game store, in-game chat, and other functionalities among the game itself.

TCG has three types of cards. Monster cards, spell cards, and environment cards. Monster cards can be summoned to one of the four monster slots on your field. The player can order the summoned monster to attack in his turn. The player can also active the monster's effects. Spell cards can be placed to one of the four spell slots on your field. The player can activate the placed spell card on your field. Environment cards can be applied to activate continuous effects. At the beginning of the game, each player's deck is randomly shuffled and is dealt five from the one's own deck. Then, the players execute actions in each of their turns. In one player's turn, the player can choose to draw one card, execute actions, and order monsters to attack.

### Summary of User Stories

#	Description
1	The User can register using a Name and a Password.
2	The User can log in to become a Player.
3	The Player can edit account information including Name, Password, and Bio.
4	The Player can create a Room with an optional Room Name and becomes a Host.
5	The Player can see a list of all Rooms with their members and Room Name.
6	The Player can join a Room by clicking on the Room Name.
7	The Player can leave the Room.
8	The Host can start a Game with people in the Room if the Room has more than 2 people agree to start the Game, all Players and Host in the Room become Gamer, and the Room is removed from the list of all Rooms.
9	The Player in a Room can refuse to start the Game and be forced to leave the Room.
10	The Gamer is dealt with 5 random cards at the beginning of the Game.
11	The Gamer has 5000 Life Points initially.
12	The Gamer can see other Players in the Game with their Name and icons indicating their cards in hands.
13	The Gamer can play in his/her turn every round and the order of turns is based on how early the Gamer originally joined the Room.

14	The Gamer may be dealt 1 random card in every round.
15	The Gamer can Summon a Marionette using a Character Card in hand.
16	The Gamer can order the Marionettes to Attack other Gamers' Marionettes and when there is no Marionette on Ground, other Gamers themselves every round.
17	The Player can be dealt 1 Character Card, 1 Spell Card, and 1 Environment Card.
18	The Gamer can Teach Marionettes compatible Spells by using Spell Cards on Marionette.
20	The Gamer can Change Environment by using an Environment Card.
24	The Gamer can be Defeated when his/her Life is below 0.
25	The Gamer receives some Spirit Points and becomes a Watcher automatically when Defeated.
26	The Gamer can Win when all the other Gamers are Defeated.
27	The Gamer receives a lot of Spirit Points and Magic Points and becomes a Player, when the Game is Ended. All Watchers become Players as well.
28	The Player can see his/her own statistics including Games Played, Games Won, Time Online, Time Played, Time Joined, and Last Seen Time.
29	The Player can chat with people in the Room.
30	The Player can be dealt more Character Cards, more Spell Cards, and more Environment Cards.
39	The Player can see a list of all Games with their members and Room Names.
40	The Player can watch a Game by clicking on the Room Name and becomes a Watcher.
41	The Watcher can see all players' Name, and icons indicating their cards in hands.
42	The Watcher can communicate with other Watcher and Gamers by posting Comments.
43	The Watcher can see Comments posted by other Watchers and Gamers.
44	The Watcher can leave the Game.
45	The Gamer can see Comments posted by Watchers and Gamers.
47	The Gamer can communicate with other Gamers and Watchers by posting Comments.
48	The Player can be dealt more Character Cards, more Spell Cards, and more Environment Cards.
49	The Player can use Spirit Points to purchase Life Upgrade which gives them extra Life in a Game.
52	The Player can view a list of all players with their Bio and status sorted in the Life Upgrade they have.
53	The Player can Follow a player in the list of all players.
54	The Player can view their Followers and their Following players in 2 lists.
57	Add environment card and integrate it within the game
58	The Player can be dealt more Character Cards, more Spell Cards, and more Environment Cards.
59	The user can see a nice UI overall.
60	The Player can tribute summon a Monster to the its Field using one or two Monster
62	The Player can see a notification when they win or lose.
67	The Player can see his/her own and other Players' statistics including the Upgrades.

## Architecture & Design

Touhou Card Game is a web-based multiplayer online card game platform which uses characters from Touhou Project. TCG's backend runs on NodeJS with MongoDB as the database while TCG's frontend is written in React. When a client is opened, it will tries to connect to the TCG server to establish a Socket.io connection. After the connection is established, the client and the server sends Socket.io messages to update states.

## Backend

TCG's backend runs on NodeJS with MongoDB as the database. We use Express to server the static resources to the frontend. We use Socket.io to establish the two-way communication channel. We use mongoose as the database driver. Essentially, the server maintains user sessions, rooms, and games using several simple functions. The `Game` class contains information about a particular game. One `Game` object has several `Player` objects, which each has one `Field` object holding `Card` objects. All special `Card` classes are inherited from the `Card` class. For example, the `MonsterCard` class is inherited from the `Card` class. It adds properties that are specific to the monster cards to the `Card` class. All monster cards are derived classes from the `MonsterCard` class.

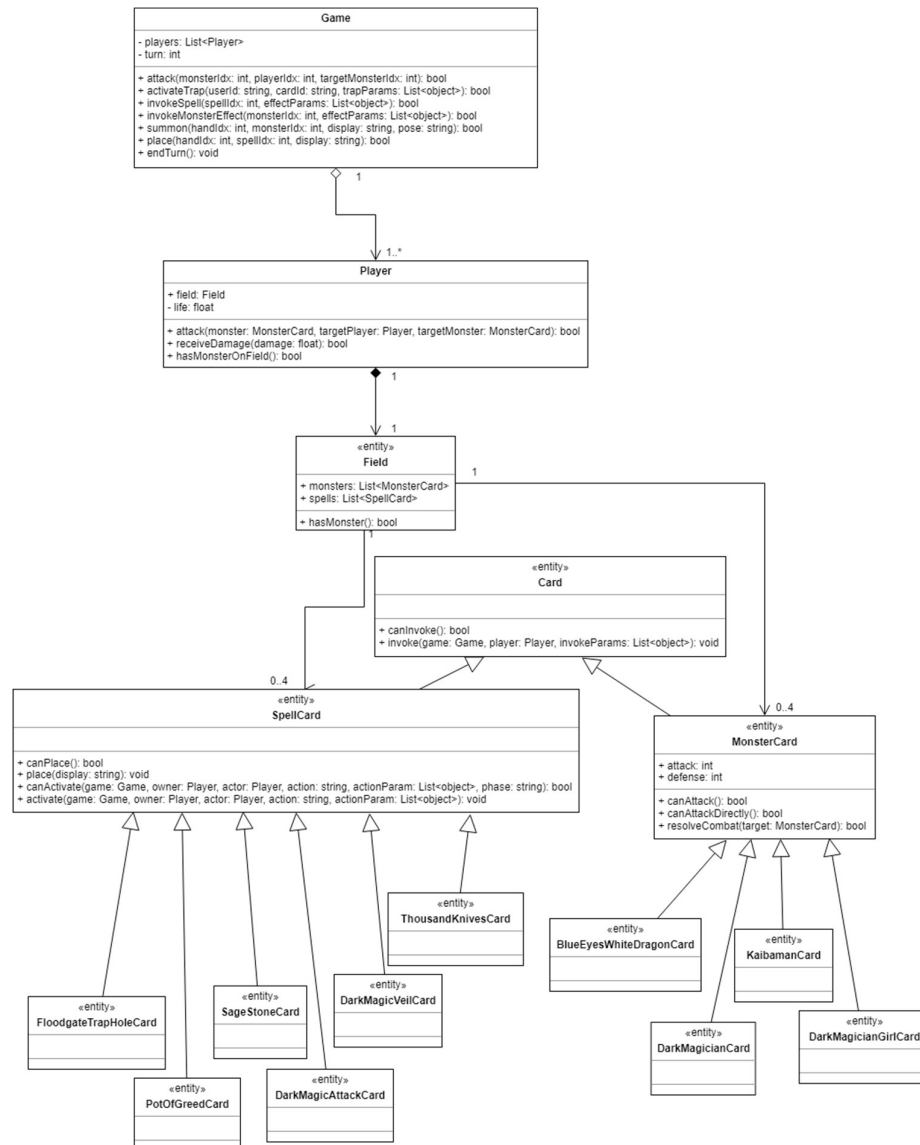


Figure 6.1. The class diagram for the core system.

The `Game` class exposes all the interfaces frontend needs to perform updates on the `Game` object. When players in a room start a new game, an instance of the `Game` class is created. The players of the game is initiated from the players' information in the room.



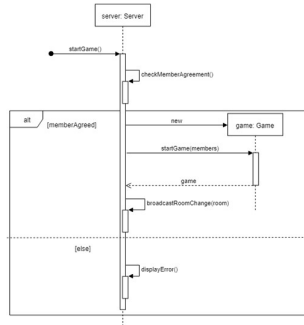


Figure 6.2. The sequence diagram for starting a game.

After the game is created, the players receive the same copy of the game whenever the internal state of a game changes. When the server receives an action from the frontend via the Socket.io connection, the server translate the said action to proper method invocations on the instance of the `Game` class held by the room. And, `Game` class may delegate some of the action to the `Player` class and `Field` class.

For example, when a monster attack action is received from the frontend, the server translate the action into an `attack()` method invocation on the instance of the `Game` class. Then, the `Game` class delegate the checks and the handling of the damage calculations to the `Player` class, `Field` class, and `MonsterCard` class.

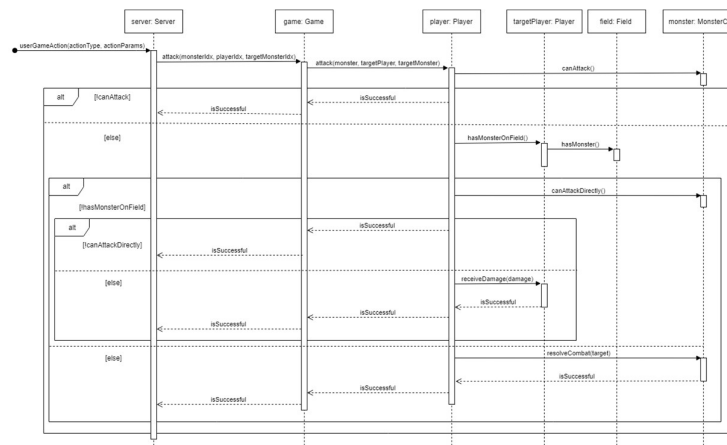


Figure 6.3. The sequence diagram for ordering the monster to attack.

In the other hand, summoning a monster or placing a spell onto the field only involves the `Player` class and the `Field` class.

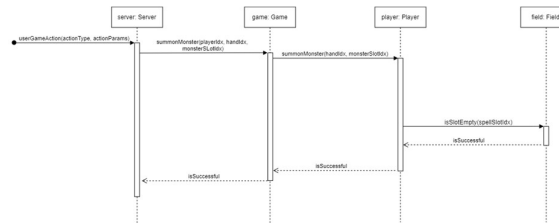


Figure 6.4. The sequence diagram for summoning a monster.

## Frontend

The frontend for TCG is designed to be simple. It is originally written in plain HTML and JavaScript for the simplicity of having no rendering frameworks. But, in the course of implementing the game's user interfaces, we found that the game interfaces are too complex to write without a frontend framework like React.

So, we rewrite the old user interfaces into React code. But, we keep the principle that the frontend is meant to be simple. For example, the frontend knows nothing about the cards. It renders the cards' actions according to the serialized data coming from the backend and sends the user's interactions back to the server as required by the server. And, the server may send back an updated version of the game that has just been changed by the user's interaction.

The frontend has seven core classes handling the rendering of TCG. The `App` class not only contains all of the web page content but also holds all of the APIs via Socket.io. Every class that needs API access must have a copy of the `App` object's reference. The `App` class also has default error handling in case of the failures of the APIs. The `Account` class handles the rendering of the account information section. The `Game` class handles the rendering of the game. The `Lobby` class handles the rendering of the lobby, the room, and the matchmaking interfaces. The `PlayersList` class handles the rendering of the list of players. The `Store` class handles the rendering of the in-game store. The `Statistics` class handles the rendering of the in-game statistics.

## Reflections and Lessons Learned

### Kailang Fu

College students are unreliable and we should account for that. We often lack professional training. We are not fully committed in getting the project done. I always account for the unreliability of the team members. I designed the project such that anyone who wants to learn JavaScript can safely join and contribute. I believe that this practice of designing and executing the project expecting risks and challenges has led to the success of this project.

### Ellen Ni

I think that it is important to divide work clearly at the beginning of each iteration and set deadlines early so that the individual knows what they should be doing and the team knows when it will be done. It was easier to do work when we were all together because I could ask people for help. I was also more motivated because I could see people working around me.

### Kuo Bao

It was my first time that I started a new project with an unfamiliar language. This experience was both challenging and interesting. I definitely learned a lot from my teammates and really appreciated their help and hard working for this project. Frankly speaking, I could have committed myself more to this project and I did not help much with the front end of our project. Anyway, thanks to my team, I had fun and learned a lot.

### Bingxu Han

I might be the one with the weakest CS background in our team and know nothing about the front end and Javascript at the start of the team. Learning completely new things by myself might be challenging at first but making contributions in this project gave me solid confidence when facing new challenges in the future.

### Charlie Yang

Doing another project with javascript is what I always wanted. Yu-Gi-Oh! Has been one of the favorite games of all time so I was not hesitated at all to jump on board. However, working with other college students

has a lot of risks such as inconsistent meeting schedule, different programming skills. It was the first time I realized that planning ahead is so much important in a group project than in a personal project. If we don't do any planning, the project will face many issues. Fortunately, thanks to the contents we learned in 428, we have learned many software engineering methods to maximize skills from each person, and thusly we are able to finish this big project on time. In the future, I will apply this method learned from doing this project to real-world projects. It was really fun doing this project.

## Zhenglun Chen

Yu-Gi-Oh! has always been one of my favorite games. When I learned that this project is about to make something similar, I was quite excited. However, when it came to actually building the project, as I don't have any prior knowledge in JavaScript, I struggled a lot during the first several weeks. Honestly, the most part I wrote are the cards and tests, and if I had experience in full-stack development, I would have helped more on the general framework of this project. Throughout the past semester, I learned a lot and had great fun. I want to thank my team members for their endeavor and patience, which have led to the eventual success of this project!

# Appendix

## Classes

Field

Field

Game

Game

Player

Player

## Field

Field

Kind: global class

- `Field`
  - `.monsterSlots` : `Array.<MonsterCard>`
  - `.spellSlots` : `Array.<SpellCard>`
  - `.environmentSlot` : `EnvironmentCard`
  - `.graveyard` : `Array.<Card>`
  - `.oblivion` : `Array.<Card>`
  - `.slotIds` : `Array.<String>`
  - `.hasMonsterSlot(id)` : `Boolean`
  - `.hasSpellSlot(id)` : `Boolean`
  - `.isSlotEmpty(id)` : `Boolean`
  - `.setSlot(id, card)` : `Card`
  - `.getSlot(id)` : `Card`
  - `.getMonsterSlotId(slotIndex)` : `String`
  - `.getSpellSlotId(slotIndex)` : `String`
  - `.removeCardFromOblivion(index)` : `Card`
  - `.endTurn()`
  - `.hasMonster()` : `Boolean`
  - `.findSpellById(id)` : `SpellCard | null`
  - `.findMonsterById(id)` : `MonsterCard | null`
  - `.findCardById(id)` : `Card`
  - `.removeCardById(id)` : `Object`
  - `.killMonsterById(id)`
  - `.killSpellById(id)`

`field.monsterSlots` : `Array.<MonsterCard>`

Kind: instance property of `Field`

**field.spellSlots** : Array.<SpellCard>

**Kind:** instance property of **Field**

**field.environmentSlot** : EnvironmentCard

**Kind:** instance property of **Field**

**field.graveyard** : Array.<Card>

**Kind:** instance property of **Field**

**field.oblivion** : Array.<Card>

**Kind:** instance property of **Field**

**field.slotIds** : Array.<String>

**Kind:** instance property of **Field**

**field.hasMonsterSlot(id)** : Boolean

Check if has monster slot with id

**Kind:** instance method of **Field**

Param

id

Type

String

**field.hasSpellSlot(id)** : Boolean

Check if has spell slot with id

**Kind:** instance method of **Field**

Param

id

Type

String

**field.isSlotEmpty(id)** : Boolean

Check if slot empty

**Kind:** instance method of **Field**

Param

id

Type

String

**field.setSlot(id, card)** : Card

Set the slot

**Kind:** instance method of **Field**

Param

id

card

Type

String

Card

**field.getSlot(id) : Card**

Get slot

**Kind:** instance method of `Field`

Param	Type
id	String

**field.getMonsterSlotId(slotIndex) : String**

Get monster slot ID

**Kind:** instance method of `Field`

Param	Type
slotIndex	Number

**field.getSpellSlotId(slotIndex) : String**

Get spell slot id

**Kind:** instance method of `Field`

Param	Type
slotIndex	Number

**field.removeCardFromOblivion(index) : Card**

Remove a card from oblivion field

**Kind:** instance method of `Field`

**Returns:** Card - card

Param	Type	Description
index	Number	card index in oblivion field

**field.endTurn()**

end turn

**Kind:** instance method of `Field`

**field.hasMonster() : Boolean**

Check if has monster

**Kind:** instance method of `Field`

**field.findSpellById(id) : SpellCard | null**

**Kind:** instance method of `Field`

Param	Type
id	String

`field.findMonsterById(id) : MonsterCard | null`

Kind: instance method of `Field`

Param	Type
<code>id</code>	<code>String</code>

`field.findCardById(id) : Card`

Find card by id

Kind: instance method of `Field`

Param	Type
<code>id</code>	<code>String</code>

`field.removeCardById(id) : Object`

Remove card by id

Kind: instance method of `Field`

Param	Type
<code>id</code>	<code>String</code>

`field.killMonsterById(id)`

Kill monster by id

Kind: instance method of `Field`

Param	Type
<code>id</code>	<code>String</code>

`field.killSpellById(id)`

Kill spell by id

Kind: instance method of `Field`

Param	Type
<code>id</code>	<code>String</code>

## `Game`

Game

Kind: global class

- `Game`
  - `new Game(users)`
  - *instance*
    - `.players : Array.<Player>`
    - `.round : Number`
    - `.turn : Number`
    - `.playerIndexById : Object.<string, number>`
    - `.isMyTurn(userId) : Boolean`

- `.invokeSpell(spellId, invokeParams) : any`
- `.invokeMonsterEffect(monsterId, invokeParams) : any`
- `.draw() : Object`
- `.summon(monsterId, slotId, display, pose, tributes) : Object`
- `.place(spellId, slotId, display) : Object`
- `.applyEnvironment(envId) : Object | Object`
- `.changeDisplay(monsterId, display) : Object`
- `.changePose(monsterId, pose) : Object`
- `.attack(monsterId, targetMonsterId) : Object`
- `.directAttack(monsterId, targetPlayerId) : Object`
- `.endTurn() : Object`
- `.checkGameEnd() : boolean`
- `.takeSnapshot() : Object`
- `.findCardOwnerById(cardId) : Player | null`
- `.findPlayer(userId) : Player | null`
- *static*
  - `.success() : Object`
  - `.error(msg) : Object`

#### `new Game(users)`

Param	Type	Description
users	Array.<any>	users of a game

#### `game.players : Array.<Player>`

**Kind:** instance property of `Game`

#### `game.round : Number`

**Kind:** instance property of `Game`

#### `game.turn : Number`

**Kind:** instance property of `Game`

#### `game.playerIndexById : Object.<string, number>`

**Kind:** instance property of `Game`

#### `game.isMyTurn(userId) : Boolean`

check if is my turn

**Kind:** instance method of `Game`

Param	Type
-------	------



userId	String
--------	--------

**game.invokeSpell(spellId, invokeParams) : any**

invoke spell

**Kind:** instance method of [Game](#)

Param	Type
spellId	String
invokeParams	Array.<String>

**game.invokeMonsterEffect(monsterId, invokeParams) : any**

invoke monster effect

**Kind:** instance method of [Game](#)

Param	Type
monsterId	String
invokeParams	Array.<String>

**game.draw() : Object**

draw a card

**Kind:** instance method of [Game](#)

**Returns:** Object - error message

**game.summon(monsterId, slotId, display, pose, tributes) : Object**

normal summon a monster

**Kind:** instance method of [Game](#)

**Returns:** Object - error message

Param	Type	Description
monsterId	String	card index in hand
slotId	String	card index in monsterSlots
display	String	card display
pose	String	card pose
tributes	Array.<String>	tributes for the summon

**game.place(spellId, slotId, display) : Object**

place spell

**Kind:** instance method of [Game](#)

Param	Type
spellId	String
slotId	String

display	String
---------	--------

```
game.applyEnvironment(envId) : Object | Object
```

apply environment card to field, replace existing one

**Kind:** instance method of [Game](#)

Param	Type	Description
envId	string	id of the environment card

`game.changeDisplay(monsterId, display) : Object`

change display

**Kind:** instance method of [Game](#)

Param	Type
monsterId	String
display	String

`game.changePose(monsterId, pose) : Object`

change pose

**Kind:** instance method of [Game](#)

Param	Type
monsterId	String
pose	String

`game.attack(monsterId, targetMonsterId) : Object`

## Attack

**Kind:** instance method of [Game](#)

Param	Type
monsterId	String
targetMonsterId	String

```
game.directAttack(monsterId, targetPlayerId) : Object
```

direct attack

**Kind:** instance method of [Game](#)

Param	Type
monsterId	String
targetPlayerId	String

`game.endTurn()` : Object

end turn

**Kind:** instance method of `Game`  
**Returns:** Object - error message

`game.checkGameEnd() : boolean`

Check if game end

**Kind:** instance method of `Game`

`game.takeSnapshot() : Object`

take snapshot

**Kind:** instance method of `Game`

**Returns:** Object - the snapshot of the game

`game.findCardOwnerById(cardId) : Player | null`

find card owner by id

**Kind:** instance method of `Game`

**Returns:** `Player` | null - card

Param	Type
cardId	String

`game.findPlayer(userId) : Player | null`

find player

**Kind:** instance method of `Game`

**Returns:** `Player` | null - player

Param	Type
userId	String

`Game.success() : Object`

Create game response

**Kind:** static method of `Game`

`Game.error(msg) : Object`

Create game response

**Kind:** static method of `Game`

Param	Type
msg	String

`Player`

Player

**Kind:** global class

- `Player`

- `new Player(user)`
- `.deck : Array.<Card>`
- `.hand : Array.<Card>`
- `.hasActivated : Object.<string, boolean>`
- `.canDraw() : Boolean`
- `.draw()`
- `.endTurn()`
- `.findCardInDeckByName(name) : Number`
- `.removeCardFromDeck(index) : Card`
- `.canBeDirectlyAttacked() : Boolean`
- `.directAttack(monster, targetUser)`
- `.attack(monster, targetUser, targetMonster)`
- `.receiveDamage(attack)`
- `.removeCardInHand(index) : Card`
- `.findCardInHandById(id) : Card | null`
- `.removeCardInHandById(id)`

#### `new Player(user)`

Param	Type	Description
<code>user</code>	<code>any</code>	the user

#### `player.deck : Array.<Card>`

**Kind:** instance property of `Player`

#### `player.hand : Array.<Card>`

**Kind:** instance property of `Player`

#### `player.hasActivated : Object.<string, boolean>`

**Kind:** instance property of `Player`

#### `player.canDraw() : Boolean`

Check if can draw

**Kind:** instance method of `Player`

#### `player.draw()`

Draw a card.

**Kind:** instance method of `Player`

#### `player.endTurn()`

End turn

**Kind:** instance method of [Player](#)

[player.findCardInDeckByName\(name\)](#) : [Number](#)

**Kind:** instance method of [Player](#)

**Returns:** [Number](#) - index

Param	Type
name	string

[player.removeCardFromDeck\(index\)](#) : [Card](#)

**Kind:** instance method of [Player](#)

Param	Type	Description
index	Number	of card in Deck

[player.canBeDirectlyAttacked\(\)](#) : [Boolean](#)

Check if can be directly attacked

**Kind:** instance method of [Player](#)

[player.directAttack\(monster, targetUser\)](#)

Direct attack

**Kind:** instance method of [Player](#)

Param	Type
monster	MonsterCard
targetUser	<a href="#">Player</a>

[player.attack\(monster, targetUser, targetMonster\)](#)

**Kind:** instance method of [Player](#)

Param	Type
monster	MonsterCard
targetUser	<a href="#">Player</a>
targetMonster	MonsterCard

[player.receiveDamage\(attack\)](#)

Receive damage

**Kind:** instance method of [Player](#)

Param	Type
attack	Number

[player.removeCardInHand\(index\)](#) : [Card](#)

Remove a card from hand

**Kind:** instance method of [Player](#)

**Returns:** Card - card

Param	Type	Description
index	Number	card index in hand

[player.findCardInHandById\(id\) : Card | null](#)

**Kind:** instance method of [Player](#)

Param	Type	Description
id	String	card id

[player.removeCardInHandById\(id\)](#)

**Kind:** instance method of [Player](#)

Param	Type	Description
id	String	card id

## Classes

BlueEyesWhiteDragonCard - MonsterCard

Blue-Eyes White Dragon Card

Card

Card

DarkMagicAttackCard - SpellCard

DarkMagicAttackCard

DarkMagicianCard - MonsterCard

Dark Magician Card

DarkMagicianGirlCard - MonsterCard

DarkMagicianGirl Card

DarkMagicVeilCard - SpellCard

DarkMagicVeilCard

EnvironmentCard - Card

EnvironmentCard

FirestormMonarchCard - MonsterCard

FirestormMonarch Card

KaibamanCard - MonsterCard

Kaibaman Card

MobiusTheFrostMonarchCard - MonsterCard

MobiusTheFrostMonarchCard

MonsterCard - Card

Monster Card

PotOfGreedCard - SpellCard

PotOfGreedCard

RaizaTheStormMonarchCard - MonsterCard  
RaizaTheStormMonarchCard  
SageStoneCard - SpellCard  
SageStoneCard  
SorcerousSpellWallCard - EnvironmentCard  
SorcerousSpellWallCard  
SpellbookOfEternityCard - SpellCard  
SpellbookOfEternityCard  
SpellbookOfSecretsCard - SpellCard  
SpellbookOfSecretsCard  
ThousandKnivesCard - SpellCard  
ThousandKnivesCard  
TwistedSpaceCard - EnvironmentCard  
TwistedSpaceCard  
ZaborgTheThunderMonarchCard - MonsterCard  
ZaborgTheThunderMonarchCard

#### BlueEyesWhiteDragonCard - MonsterCard

Blue-Eyes White Dragon Card

**Kind:** global class

**Extends:** [MonsterCard](#)

- [BlueEyesWhiteDragonCard - MonsterCard](#)
  - [.canSummon\(display, pose\)](#) : Boolean
  - [.summon\(display, pose\)](#)
  - [.canChangeDisplay\(display\)](#) : Boolean
  - [.canChangePose\(pose\)](#) : Boolean
  - [.changeDisplay\(display\)](#)
  - [.changePose\(pose\)](#)
  - [.canAttack\(\)](#) : Boolean
  - [.attack\(\)](#)
  - [.endTurn\(\)](#)
  - [.takeSnapshot\(\)](#) : Object
  - [.canInvoke\(game, player, invokeParams\)](#) : Boolean
  - [.invoke\(game, player, invokeParams\)](#)
  - [.canPlace\(display\)](#) : Boolean
  - [.place\(display\)](#)

[blueEyesWhiteDragonCard.canSummon\(display, pose\)](#) : Boolean

check if can summon

**Kind:** instance method of [BlueEyesWhiteDragonCard](#)

Param	Type
display	String
pose	String

**blueEyesWhiteDragonCard.summon(display, pose)**

summon

**Kind:** instance method of [BlueEyesWhiteDragonCard](#)

Param	Type
display	String
pose	String

**blueEyesWhiteDragonCard.canChangeDisplay(display) : Boolean**

check if can change display

**Kind:** instance method of [BlueEyesWhiteDragonCard](#)

Param	Type
display	String

**blueEyesWhiteDragonCard.canChangePose(pose) : Boolean**

check if can change pose

**Kind:** instance method of [BlueEyesWhiteDragonCard](#)

Param	Type
pose	String

**blueEyesWhiteDragonCard.changeDisplay(display)**

change display

**Kind:** instance method of [BlueEyesWhiteDragonCard](#)

Param	Type
display	String

**blueEyesWhiteDragonCard.changePose(pose)**

change pose

**Kind:** instance method of [BlueEyesWhiteDragonCard](#)

Param	Type
pose	String

**blueEyesWhiteDragonCard.canAttack() : Boolean**

check can attack

**Kind:** instance method of [BlueEyesWhiteDragonCard](#)



**blueEyesWhiteDragonCard.attack()**

attack

**Kind:** instance method of [BlueEyesWhiteDragonCard](#)

**blueEyesWhiteDragonCard.endTurn()**

end turn

**Kind:** instance method of [BlueEyesWhiteDragonCard](#)

**blueEyesWhiteDragonCard.takeSnapshot() : Object**

take snapshot

**Kind:** instance method of [BlueEyesWhiteDragonCard](#)

**blueEyesWhiteDragonCard.canInvoke(game, player, invokeParams) : Boolean**

Check if can invoke

**Kind:** instance method of [BlueEyesWhiteDragonCard](#)

Param	Type
game	Game
player	Player
invokeParams	Array

**blueEyesWhiteDragonCard.invoke(game, player, invokeParams)**

Invoke card effects

**Kind:** instance method of [BlueEyesWhiteDragonCard](#)

Param	Type
game	Game
player	Player
invokeParams	Array

**blueEyesWhiteDragonCard.canPlace(display) : Boolean**

Check if can place

**Kind:** instance method of [BlueEyesWhiteDragonCard](#)

Param	Type
display	String

**blueEyesWhiteDragonCard.place(display)**

Place card on ground

**Kind:** instance method of [BlueEyesWhiteDragonCard](#)

Param	Type
display	String

## Card

Card

**Kind:** global class

- `Card`
  - `new Card(name, desc, imgUrl)`
  - *instance*
    - `.endTurn()`
    - `.canInvoke(game, player, invokeParams) : Boolean`
    - `.invoke(game, player, invokeParams)`
    - `.canSummon(display, pose) : Boolean`
    - `.summon(display, pose)`
    - `.canPlace(display) : Boolean`
    - `.place(display)`
    - `.takeSnapshot() : Object`
  - *static*
    - `.generateId(len) : String`
    - `.createAction(name, desc, position, params) : Object`
    - `.createActionParam(select, position, owner, desc) : Object`

### `new Card(name, desc, imgUrl)`

Param	Type	Description
name	String	unique name
desc	String	description
imgUrl	String	url to img of the card

### `card.endTurn()`

End turn.

**Kind:** instance method of `Card`

### `card.canInvoke(game, player, invokeParams) : Boolean`

Check if can invoke

**Kind:** instance method of `Card`

Param	Type
game	Game
player	Player
invokeParams	Array

### `card.invoke(game, player, invokeParams)`

Invoke card effects

**Kind:** instance method of `Card`

Param	Type
game	Game
player	Player
invokeParams	Array

#### `card.canSummon(display, pose) : Boolean`

Check if can summon

**Kind:** instance method of `Card`

Param	Type
display	String
pose	String

#### `card.summon(display, pose)`

Summon

**Kind:** instance method of `Card`

Param	Type
display	String
pose	String

#### `card.canPlace(display) : Boolean`

Check if can place

**Kind:** instance method of `Card`

Param	Type
display	String

#### `card.place(display)`

Place card on ground

**Kind:** instance method of `Card`

Param	Type
display	String

#### `card.takeSnapshot() : Object`

{actions: [{name, stage, in, params: [select, in, of]]}] select: Game.PLAYER  
userId select: Game.SLOT slotId select: Game.CARD cardId

**Kind:** instance method of `Card`

**Returns:** Object - snapshot

#### `Card.generateId(len) : String`

Length of the generated ID

**Kind:** static method of `Card`

Param	Type
len	Number

**Card.createAction(name, desc, position, params) : Object**

Create an action descriptor for snapshot

**Kind:** static method of [Card](#)

Param	Type
name	String
desc	String
position	String
params	Array

**Card.createActionParam(select, position, owner, desc) : Object**

Create an action parameter descriptor for snapshot

**Kind:** static method of [Card](#)

Param	Type
select	String
position	String
owner	String
desc	String

**DarkMagicAttackCard - SpellCard**

DarkMagicAttackCard

**Kind:** global class

**Extends:** SpellCard

- [DarkMagicAttackCard - SpellCard](#)
  - [.canInvoke\(game, player\) : Boolean](#)
  - [.invoke\(game, player, invokeParams\)](#)
  - [.takeSnapshot\(\) : Object](#)
  - [.canPlace\(\) : Boolean](#)
  - [.place\(display\)](#)

**darkMagicAttackCard.canInvoke(game, player) : Boolean**

Check if can invoke

**Kind:** instance method of [DarkMagicAttackCard](#)

**Overrides:** [canInvoke](#)

Param	Type
game	Game
player	Player

`darkMagicAttackCard.invoke(game, player, invokeParams)`

Invoke card effects

**Kind:** instance method of `DarkMagicAttackCard`

Param	Type	Description
game	Game	
player	Player	
invokeParams	Array.<String>	[opponentId]

`darkMagicAttackCard.takeSnapshot() : Object`

Take snapshot

**Kind:** instance method of `DarkMagicAttackCard`

**Overrides:** `takeSnapshot`

`darkMagicAttackCard.canPlace() : Boolean`

Check if can place

**Kind:** instance method of `DarkMagicAttackCard`

`darkMagicAttackCard.place(display)`

Place card on ground

**Kind:** instance method of `DarkMagicAttackCard`

Param	Type
display	String

**DarkMagicianCard - MonsterCard**

Dark Magician Card

**Kind:** global class

**Extends:** `MonsterCard`

- `DarkMagicianCard - MonsterCard`
  - `.canSummon(display, pose) : Boolean`
  - `.summon(display, pose)`
  - `.canChangeDisplay(display) : Boolean`
  - `.canChangePose(pose) : Boolean`
  - `.changeDisplay(display)`
  - `.changePose(pose)`
  - `.canAttack() : Boolean`
  - `.attack()`
  - `.endTurn()`
  - `.takeSnapshot() : Object`
  - `.canInvoke(game, player, invokeParams) : Boolean`
  - `.invoke(game, player, invokeParams)`

- `.canPlace(display) : Boolean`
- `.place(display)`

`darkMagicianCard.canSummon(display, pose) : Boolean`

check if can summon

**Kind:** instance method of `DarkMagicianCard`

Param	Type
display	String
pose	String

`darkMagicianCard.summon(display, pose)`

summon

**Kind:** instance method of `DarkMagicianCard`

Param	Type
display	String
pose	String

`darkMagicianCard.canChangeDisplay(display) : Boolean`

check if can change display

**Kind:** instance method of `DarkMagicianCard`

Param	Type
display	String

`darkMagicianCard.canChangePose(pose) : Boolean`

check if can change pose

**Kind:** instance method of `DarkMagicianCard`

Param	Type
pose	String

`darkMagicianCard.changeDisplay(display)`

change display

**Kind:** instance method of `DarkMagicianCard`

Param	Type
display	String

`darkMagicianCard.changePose(pose)`

change pose

**Kind:** instance method of `DarkMagicianCard`

Param	Type

pose String

**darkMagicianCard.canAttack() : Boolean**

check can attack

**Kind:** instance method of [DarkMagicianCard](#)

**darkMagicianCard.attack()**

attack

**Kind:** instance method of [DarkMagicianCard](#)

**darkMagicianCard.endTurn()**

end turn

**Kind:** instance method of [DarkMagicianCard](#)

**darkMagicianCard.takeSnapshot() : Object**

take snapshot

**Kind:** instance method of [DarkMagicianCard](#)

**darkMagicianCard.canInvoke(game, player, invokeParams) : Boolean**

Check if can invoke

**Kind:** instance method of [DarkMagicianCard](#)

Param	Type
game	Game
player	Player
invokeParams	Array

**darkMagicianCard.invoke(game, player, invokeParams)**

Invoke card effects

**Kind:** instance method of [DarkMagicianCard](#)

Param	Type
game	Game
player	Player
invokeParams	Array

**darkMagicianCard.canPlace(display) : Boolean**

Check if can place

**Kind:** instance method of [DarkMagicianCard](#)

Param	Type
display	String

### `darkMagicianCard.place(display)`

Place card on ground

**Kind:** instance method of `DarkMagicianCard`

Param	Type
display	String

### `DarkMagicianGirlCard - MonsterCard`

DarkMagicianGirl Card

**Kind:** global class

**Extends:** `MonsterCard`

- `DarkMagicianGirlCard - MonsterCard`
  - `.canInvoke()` : Boolean
  - `.invoke(game, player)`
  - `.takeSnapshot()` : Object
  - `.canSummon(display, pose)` : Boolean
  - `.summon(display, pose)`
  - `.canChangeDisplay(display)` : Boolean
  - `.canChangePose(pose)` : Boolean
  - `.changeDisplay(display)`
  - `.changePose(pose)`
  - `.canAttack()` : Boolean
  - `.attack()`
  - `.endTurn()`
  - `.canPlace(display)` : Boolean
  - `.place(display)`

### `darkMagicianGirlCard.canInvoke()` : Boolean

Check if can invoke

**Kind:** instance method of `DarkMagicianGirlCard`

**Overrides:** `canInvoke`

### `darkMagicianGirlCard.invoke(game, player)`

Invoke card effects

**Kind:** instance method of `DarkMagicianGirlCard`

**Overrides:** `invoke`

Param	Type
game	Game
player	Player



**darkMagicianGirlCard.takeSnapshot() : Object**

Take snapshot

**Kind:** instance method of [DarkMagicianGirlCard](#)

**Overrides:** [takeSnapshot](#)

**darkMagicianGirlCard.canSummon(display, pose) : Boolean**

check if can summon

**Kind:** instance method of [DarkMagicianGirlCard](#)

Param	Type
display	String
pose	String

**darkMagicianGirlCard.summon(display, pose)**

summon

**Kind:** instance method of [DarkMagicianGirlCard](#)

Param	Type
display	String
pose	String

**darkMagicianGirlCard.canChangeDisplay(display) : Boolean**

check if can change display

**Kind:** instance method of [DarkMagicianGirlCard](#)

Param	Type
display	String

**darkMagicianGirlCard.canChangePose(pose) : Boolean**

check if can change pose

**Kind:** instance method of [DarkMagicianGirlCard](#)

Param	Type
pose	String

**darkMagicianGirlCard.changeDisplay(display)**

change display

**Kind:** instance method of [DarkMagicianGirlCard](#)

Param	Type
display	String

**darkMagicianGirlCard.changePose(pose)**

change pose

**Kind:** instance method of [DarkMagicianGirlCard](#)

Param	Type
pose	String

**darkMagicianGirlCard.canAttack() : Boolean**

check can attack

**Kind:** instance method of [DarkMagicianGirlCard](#)

**darkMagicianGirlCard.attack()**

attack

**Kind:** instance method of [DarkMagicianGirlCard](#)

**darkMagicianGirlCard.endTurn()**

end turn

**Kind:** instance method of [DarkMagicianGirlCard](#)

**darkMagicianGirlCard.canPlace(display) : Boolean**

Check if can place

**Kind:** instance method of [DarkMagicianGirlCard](#)

Param	Type
display	String

**darkMagicianGirlCard.place(display)**

Place card on ground

**Kind:** instance method of [DarkMagicianGirlCard](#)

Param	Type
display	String

**DarkMagicVeilCard - SpellCard**

DarkMagicVeilCard

**Kind:** global class

**Extends:** SpellCard

- [DarkMagicVeilCard](#) - SpellCard
  - [.canInvoke\(game, player, invokeParams\) : Boolean](#)
  - [.invoke\(game, player, invokeParams\)](#)
  - [.takeSnapshot\(\) : Object](#)
  - [.canPlace\(\) : Boolean](#)
  - [.place\(display\)](#)

**darkMagicVeilCard.canInvoke(game, player, invokeParams) : Boolean**

**Kind:** instance method of [DarkMagicVeilCard](#)

**Overrides:** [canInvoke](#)

Param	Type	Description
game	Game	
player	Player	
invokeParams	Array.<String>	[slotId]

**darkMagicVeilCard.invoke(game, player, invokeParams)**

**Kind:** instance method of [DarkMagicVeilCard](#)

Param	Type	Description
game	Game	
player	Player	
invokeParams	Array.<String>	[slotId]

**darkMagicVeilCard.takeSnapshot() : Object**

Take snapshot

**Kind:** instance method of [DarkMagicVeilCard](#)

**Overrides:** [takeSnapshot](#)

**darkMagicVeilCard.canPlace() : Boolean**

Check if can place

**Kind:** instance method of [DarkMagicVeilCard](#)

**darkMagicVeilCard.place(display)**

Place card on ground

**Kind:** instance method of [DarkMagicVeilCard](#)

Param	Type
display	String

## **EnvironmentCard - Card**

EnvironmentCard

**Kind:** global class

**Extends:** [Card](#)

- [EnvironmentCard - Card](#)
  - [new EnvironmentCard\(name, desc, imgUrl\)](#)
  - [.place\(display\)](#)
  - [.takeSnapshot\(\) : Object](#)
  - [.endTurn\(\)](#)
  - [.canInvoke\(game, player, invokeParams\) : Boolean](#)

- `.invoke(game, player, invokeParams)`
- `.canSummon(display, pose) : Boolean`
- `.summon(display, pose)`
- `.canPlace(display) : Boolean`

**`new EnvironmentCard(name, desc, imgUrl)`**

Param	Type	Description
name	string	name of the environment card
desc	string	description of the card
imgUrl	string	imageUrl of the card

**`environmentCard.place(display)`**

set the display of environment card

**Kind:** instance method of `EnvironmentCard`

**Overrides:** `place`

Param	Type
display	string

**`environmentCard.takeSnapshot() : Object`**

Take snapshot

**Kind:** instance method of `EnvironmentCard`

**Overrides:** `takeSnapshot`

**`environmentCard.endTurn()`**

End turn.

**Kind:** instance method of `EnvironmentCard`

**`environmentCard.canInvoke(game, player, invokeParams) : Boolean`**

Check if can invoke

**Kind:** instance method of `EnvironmentCard`

Param	Type
game	Game
player	Player
invokeParams	Array

**`environmentCard.invoke(game, player, invokeParams)`**

Invoke card effects

**Kind:** instance method of `EnvironmentCard`

Param	Type
game	Game
player	Player
invokeParams	Array

#### `environmentCard.canSummon(display, pose) : Boolean`

Check if can summon

**Kind:** instance method of `EnvironmentCard`

Param	Type
display	String
pose	String

#### `environmentCard.summon(display, pose)`

Summon

**Kind:** instance method of `EnvironmentCard`

Param	Type
display	String
pose	String

#### `environmentCard.canPlace(display) : Boolean`

Check if can place

**Kind:** instance method of `EnvironmentCard`

Param	Type
display	String

#### `FirestormMonarchCard - MonsterCard`

FirestormMonarch Card

**Kind:** global class

**Extends:** `MonsterCard`

- `FirestormMonarchCard - MonsterCard`
  - `.canInvoke() : Boolean`
  - `.invoke(game, player, invokeParams)`
  - `.takeSnapshot() : Object`
  - `.canSummon(display, pose) : Boolean`
  - `.summon(display, pose)`
  - `.canChangeDisplay(display) : Boolean`
  - `.canChangePose(pose) : Boolean`
  - `.changeDisplay(display)`
  - `.changePose(pose)`
  - `.canAttack() : Boolean`

- .attack()
- .endTurn()
- .canPlace(display) : Boolean
- .place(display)

#### `firestormMonarchCard.canInvoke() : Boolean`

Check if can invoke

**Kind:** instance method of `FirestormMonarchCard`

**Overrides:** `canInvoke`

#### `firestormMonarchCard.invoke(game, player, invokeParams)`

invoke power

**Kind:** instance method of `FirestormMonarchCard`

**Overrides:** `invoke`

Param	Type	Description
game	Game	
player	Player	
invokeParams	Array.<String>	an array of 4 numbers 1st number is the player's index

#### `firestormMonarchCard.takeSnapshot() : Object`

Take snapshot

**Kind:** instance method of `FirestormMonarchCard`

**Overrides:** `takeSnapshot`

#### `firestormMonarchCard.canSummon(display, pose) : Boolean`

check if can summon

**Kind:** instance method of `FirestormMonarchCard`

Param	Type
display	String
pose	String

#### `firestormMonarchCard.summon(display, pose)`

summon

**Kind:** instance method of `FirestormMonarchCard`

Param	Type
display	String
pose	String

**firestormMonarchCard.canChangeDisplay(display) : Boolean**

check if can change display

**Kind:** instance method of [FirestormMonarchCard](#)

Param	Type
display	String

**firestormMonarchCard.canChangePose(pose) : Boolean**

check if can change pose

**Kind:** instance method of [FirestormMonarchCard](#)

Param	Type
pose	String

**firestormMonarchCard.changeDisplay(display)**

change display

**Kind:** instance method of [FirestormMonarchCard](#)

Param	Type
display	String

**firestormMonarchCard.changePose(pose)**

change pose

**Kind:** instance method of [FirestormMonarchCard](#)

Param	Type
pose	String

**firestormMonarchCard.canAttack() : Boolean**

check can attack

**Kind:** instance method of [FirestormMonarchCard](#)

**firestormMonarchCard.attack()**

attack

**Kind:** instance method of [FirestormMonarchCard](#)

**firestormMonarchCard.endTurn()**

end turn

**Kind:** instance method of [FirestormMonarchCard](#)

**firestormMonarchCard.canPlace(display) : Boolean**

Check if can place

**Kind:** instance method of [FirestormMonarchCard](#)

Param	Type
display	String

#### `firestormMonarchCard.place(display)`

Place card on ground

**Kind:** instance method of `FirestormMonarchCard`

Param	Type
display	String

#### `KaibamanCard - MonsterCard`

Kaibaman Card

**Kind:** global class

**Extends:** `MonsterCard`

- `KaibamanCard - MonsterCard`
  - `.canInvoke(game, player) : Boolean`
  - `.invoke(game, player)`
  - `.takeSnapshot() : Object`
  - `.canSummon(display, pose) : Boolean`
  - `.summon(display, pose)`
  - `.canChangeDisplay(display) : Boolean`
  - `.canChangePose(pose) : Boolean`
  - `.changeDisplay(display)`
  - `.changePose(pose)`
  - `.canAttack() : Boolean`
  - `.attack()`
  - `.endTurn()`
  - `.canPlace(display) : Boolean`
  - `.place(display)`

#### `kaibamanCard.canInvoke(game, player) : Boolean`

Check if can invoke

**Kind:** instance method of `KaibamanCard`

**Overrides:** `canInvoke`

Param	Type
game	Game
player	Player

#### `kaibamanCard.invoke(game, player)`

Invoke card effects



**Kind:** instance method of [KaibamanCard](#)

**Overrides:** [invoke](#)

Param	Type
game	Game
player	Player

[kaibamanCard.takeSnapshot\(\)](#) : Object

Take snapshot

**Kind:** instance method of [KaibamanCard](#)

**Overrides:** [takeSnapshot](#)

[kaibamanCard.canSummon\(display, pose\)](#) : Boolean

check if can summon

**Kind:** instance method of [KaibamanCard](#)

Param	Type
display	String
pose	String

[kaibamanCard.summon\(display, pose\)](#)

summon

**Kind:** instance method of [KaibamanCard](#)

Param	Type
display	String
pose	String

[kaibamanCard.canChangeDisplay\(display\)](#) : Boolean

check if can change display

**Kind:** instance method of [KaibamanCard](#)

Param	Type
display	String

[kaibamanCard.canChangePose\(pose\)](#) : Boolean

check if can change pose

**Kind:** instance method of [KaibamanCard](#)

Param	Type
pose	String

[kaibamanCard.changeDisplay\(display\)](#)

change display

**Kind:** instance method of [KaibamanCard](#)

Param	Type
display	String

#### `kaibamanCard.changePose(pose)`

change pose

**Kind:** instance method of `KaibamanCard`

Param	Type
pose	String

#### `kaibamanCard.canAttack() : Boolean`

check can attack

**Kind:** instance method of `KaibamanCard`

#### `kaibamanCard.attack()`

attack

**Kind:** instance method of `KaibamanCard`

#### `kaibamanCard.endTurn()`

end turn

**Kind:** instance method of `KaibamanCard`

#### `kaibamanCard.canPlace(display) : Boolean`

Check if can place

**Kind:** instance method of `KaibamanCard`

Param	Type
display	String

#### `kaibamanCard.place(display)`

Place card on ground

**Kind:** instance method of `KaibamanCard`

Param	Type
display	String

#### `MobiusTheFrostMonarchCard - MonsterCard`

MobiusTheFrostMonarchCard

**Kind:** global class

**Extends:** `MonsterCard`

- `MobiusTheFrostMonarchCard - MonsterCard`
  - `.canInvoke(game, player, invokeParams) : boolean`
  - `.invoke(game, player, invokeParams)`

- .takeSnapshot() : Object
- .canSummon(display, pose) : Boolean
- .summon(display, pose)
- .canChangeDisplay(display) : Boolean
- .canChangePose(pose) : Boolean
- .changeDisplay(display)
- .changePose(pose)
- .canAttack() : Boolean
- .attack()
- .endTurn()
- .canPlace(display) : Boolean
- .place(display)

**mobiusTheFrostMonarchCard.canInvoke(game, player, invokeParams) : boolean**

Check whether the monster can invoke his ability

**Kind:** instance method of [MobiusTheFrostMonarchCard](#)

**Overrides:** [canInvoke](#)

**Returns:** boolean - whether it can activate its ability

Param	Type	Description
game	Object	
player	Object	
invokeParams	Object	should be null

**mobiusTheFrostMonarchCard.invoke(game, player, invokeParams)**

invoke power

**Kind:** instance method of [MobiusTheFrostMonarchCard](#)

**Overrides:** [invoke](#)

Param	Type	Description
game	Game	
player	Player	
invokeParams	Array.<String>	an array of 4 numbers [0]: first cardId [1]: second cardId

**mobiusTheFrostMonarchCard.takeSnapshot() : Object**

take snapshot of the current card

**Kind:** instance method of [MobiusTheFrostMonarchCard](#)

**Overrides:** [takeSnapshot](#)

**Returns:** Object - the snapshot of the game

`mobiusTheFrostMonarchCard.canSummon(display, pose) : Boolean`

check if can summon

**Kind:** instance method of `MobiusTheFrostMonarchCard`

Param	Type
display	String
pose	String

`mobiusTheFrostMonarchCard.summon(display, pose)`

summon

**Kind:** instance method of `MobiusTheFrostMonarchCard`

Param	Type
display	String
pose	String

`mobiusTheFrostMonarchCard.canChangeDisplay(display) : Boolean`

check if can change display

**Kind:** instance method of `MobiusTheFrostMonarchCard`

Param	Type
display	String

`mobiusTheFrostMonarchCard.canChangePose(pose) : Boolean`

check if can change pose

**Kind:** instance method of `MobiusTheFrostMonarchCard`

Param	Type
pose	String

`mobiusTheFrostMonarchCard.changeDisplay(display)`

change display

**Kind:** instance method of `MobiusTheFrostMonarchCard`

Param	Type
display	String

`mobiusTheFrostMonarchCard.changePose(pose)`

change pose

**Kind:** instance method of `MobiusTheFrostMonarchCard`

Param	Type
pose	String

`mobiusTheFrostMonarchCard.canAttack() : Boolean`

check can attack

**Kind:** instance method of `MobiusTheFrostMonarchCard`

`mobiusTheFrostMonarchCard.attack()`

attack

**Kind:** instance method of `MobiusTheFrostMonarchCard`

`mobiusTheFrostMonarchCard.endTurn()`

end turn

**Kind:** instance method of `MobiusTheFrostMonarchCard`

`mobiusTheFrostMonarchCard.canPlace(display) : Boolean`

Check if can place

**Kind:** instance method of `MobiusTheFrostMonarchCard`

Param	Type
display	String

`mobiusTheFrostMonarchCard.place(display)`

Place card on ground

**Kind:** instance method of `MobiusTheFrostMonarchCard`

Param	Type
display	String

## `MonsterCard - Card`

Monster Card

**Kind:** global class

**Extends:** `Card`

- `MonsterCard - Card`
  - `new MonsterCard(name, desc, imgUrl, lv, atk, dfs)`
  - `.canSummon(display, pose) : Boolean`
  - `.summon(display, pose)`
  - `.canChangeDisplay(display) : Boolean`
  - `.canChangePose(pose) : Boolean`
  - `.changeDisplay(display)`
  - `.changePose(pose)`
  - `.canAttack() : Boolean`
  - `.attack()`
  - `.endTurn()`
  - `.takeSnapshot() : Object`
  - `.canInvoke(game, player, invokeParams) : Boolean`
  - `.invoke(game, player, invokeParams)`

- `.canPlace(display) : Boolean`
- `.place(display)`

`new MonsterCard(name, desc, imgUrl, lv, atk, dfs)`

Param	Type	Description
name	String	unique name
desc	String	description
imgUrl	String	url to img of the card
lv	Number	monster level
atk	Number	monster attack
dfs	Number	monster defense

`monsterCard.canSummon(display, pose) : Boolean`

check if can summon

**Kind:** instance method of `MonsterCard`

**Overrides:** `canSummon`

Param	Type
display	String
pose	String

`monsterCard.summon(display, pose)`

summon

**Kind:** instance method of `MonsterCard`

**Overrides:** `summon`

Param	Type
display	String
pose	String

`monsterCard.canChangeDisplay(display) : Boolean`

check if can change display

**Kind:** instance method of `MonsterCard`

Param	Type
display	String

`monsterCard.canChangePose(pose) : Boolean`

check if can change pose

**Kind:** instance method of `MonsterCard`

Param	Type
pose	String

**monsterCard.changeDisplay(display)**

change display

**Kind:** instance method of [MonsterCard](#)

Param	Type
display	String

**monsterCard.changePose(pose)**

change pose

**Kind:** instance method of [MonsterCard](#)

Param	Type
pose	String

**monsterCard.canAttack() : Boolean**

check can attack

**Kind:** instance method of [MonsterCard](#)

**monsterCard.attack()**

attack

**Kind:** instance method of [MonsterCard](#)

**monsterCard.endTurn()**

end turn

**Kind:** instance method of [MonsterCard](#)

**Overrides:** [endTurn](#)

**monsterCard.takeSnapshot() : Object**

take snapshot

**Kind:** instance method of [MonsterCard](#)

**Overrides:** [takeSnapshot](#)

**monsterCard.canInvoke(game, player, invokeParams) : Boolean**

Check if can invoke

**Kind:** instance method of [MonsterCard](#)

Param	Type
game	Game
player	Player
invokeParams	Array

**monsterCard.invoke(game, player, invokeParams)**

Invoke card effects

**Kind:** instance method of `MonsterCard`

Param	Type
game	Game
player	Player
invokeParams	Array

`monsterCard.canPlace(display) : Boolean`

Check if can place

**Kind:** instance method of `MonsterCard`

Param	Type
display	String

`monsterCard.place(display)`

Place card on ground

**Kind:** instance method of `MonsterCard`

Param	Type
display	String

`PotOfGreedCard - SpellCard`

PotOfGreedCard

**Kind:** global class

**Extends:** `SpellCard`

- `PotOfGreedCard - SpellCard`
  - `.invoke(game, player)`
  - `.takeSnapshot() : Object`
  - `.canPlace() : Boolean`
  - `.place(display)`
  - `.canInvoke(game, player, invokeParams) : Boolean`

`potOfGreedCard.invoke(game, player)`

Invoke card effects

**Kind:** instance method of `PotOfGreedCard`

Param	Type
game	Game
player	Player

`potOfGreedCard.takeSnapshot() : Object`

Take snapshot



**Kind:** instance method of `PotOfGreedCard`

**Overrides:** `takeSnapshot`

`potOfGreedCard.canPlace() : Boolean`

Check if can place

**Kind:** instance method of `PotOfGreedCard`

`potOfGreedCard.place(display)`

Place card on ground

**Kind:** instance method of `PotOfGreedCard`

Param	Type
display	String

`potOfGreedCard.canInvoke(game, player, invokeParams) : Boolean`

Check if can invoke

**Kind:** instance method of `PotOfGreedCard`

Param	Type
game	Game
player	Player
invokeParams	Array

**RaizaTheStormMonarchCard - MonsterCard**

RaizaTheStormMonarchCard

**Kind:** global class

**Extends:** `MonsterCard`

- `RaizaTheStormMonarchCard - MonsterCard`
  - `.invoke(game, player, invokeParams)`
  - `.takeSnapshot() : Object`
  - `.canSummon(display, pose) : Boolean`
  - `.summon(display, pose)`
  - `.canChangeDisplay(display) : Boolean`
  - `.canChangePose(pose) : Boolean`
  - `.changeDisplay(display)`
  - `.changePose(pose)`
  - `.canAttack() : Boolean`
  - `.attack()`
  - `.endTurn()`
  - `.canInvoke(game, player, invokeParams) : Boolean`
  - `.canPlace(display) : Boolean`
  - `.place(display)`

**raizaTheStormMonarchCard.invoke(game, player, invokeParams)**

invoke power

**Kind:** instance method of [RaizaTheStormMonarchCard](#)

**Overrides:** [invoke](#)

Param	Type	Description
game	Game	
player	Player	
invokeParams	Array.<String>	an array of 1 numbers cardId

**raizaTheStormMonarchCard.takeSnapshot() : Object**

take snapshot of the current card

**Kind:** instance method of [RaizaTheStormMonarchCard](#)

**Overrides:** [takeSnapshot](#)

**Returns:** Object - the snapshot of the game

**raizaTheStormMonarchCard.canSummon(display, pose) : Boolean**

check if can summon

**Kind:** instance method of [RaizaTheStormMonarchCard](#)

Param	Type
display	String
pose	String

**raizaTheStormMonarchCard.summon(display, pose)**

summon

**Kind:** instance method of [RaizaTheStormMonarchCard](#)

Param	Type
display	String
pose	String

**raizaTheStormMonarchCard.canChangeDisplay(display) : Boolean**

check if can change display

**Kind:** instance method of [RaizaTheStormMonarchCard](#)

Param	Type
display	String

**raizaTheStormMonarchCard.canChangePose(pose) : Boolean**

check if can change pose

**Kind:** instance method of [RaizaTheStormMonarchCard](#)

Param	Type
pose	String

**raizaTheStormMonarchCard.changeDisplay(display)**

change display

**Kind:** instance method of [RaizaTheStormMonarchCard](#)

Param	Type
display	String

**raizaTheStormMonarchCard.changePose(pose)**

change pose

**Kind:** instance method of [RaizaTheStormMonarchCard](#)

Param	Type
pose	String

**raizaTheStormMonarchCard.canAttack() : Boolean**

check can attack

**Kind:** instance method of [RaizaTheStormMonarchCard](#)

**raizaTheStormMonarchCard.attack()**

attack

**Kind:** instance method of [RaizaTheStormMonarchCard](#)

**raizaTheStormMonarchCard.endTurn()**

end turn

**Kind:** instance method of [RaizaTheStormMonarchCard](#)

**raizaTheStormMonarchCard.canInvoke(game, player, invokeParams) : Boolean**

Check if can invoke

**Kind:** instance method of [RaizaTheStormMonarchCard](#)

**Overrides:** [canInvoke](#)

Param	Type
game	Game
player	Player
invokeParams	Array

**raizaTheStormMonarchCard.canPlace(display) : Boolean**

Check if can place

**Kind:** instance method of [RaizaTheStormMonarchCard](#)

Param	Type
display	String

#### [raizaTheStormMonarchCard.place\(display\)](#)

Place card on ground

**Kind:** instance method of [RaizaTheStormMonarchCard](#)

Param	Type
display	String

#### [SageStoneCard - SpellCard](#)

SageStoneCard

**Kind:** global class

**Extends:** SpellCard

- [SageStoneCard - SpellCard](#)
  - [.canInvoke\(game, player, invokeParams\)](#) : Boolean
  - [.invoke\(game, player, invokeParams\)](#)
  - [.takeSnapshot\(\)](#) : Object
  - [.canPlace\(\)](#) : Boolean
  - [.place\(display\)](#)

#### [sageStoneCard.canInvoke\(game, player, invokeParams\) : Boolean](#)

Check if can invoke

**Kind:** instance method of [SageStoneCard](#)

**Overrides:** [canInvoke](#)

Param	Type
game	Game
player	Player
invokeParams	Array

#### [sageStoneCard.invoke\(game, player, invokeParams\)](#)

**Kind:** instance method of [SageStoneCard](#)

Param	Type
game	Game
player	Player
invokeParams	Array.<String>

#### [sageStoneCard.takeSnapshot\(\) : Object](#)

Take snapshot

**Kind:** instance method of `SageStoneCard`

**Overrides:** `takeSnapshot`

`sageStoneCard.canPlace()` : `Boolean`

Check if can place

**Kind:** instance method of `SageStoneCard`

`sageStoneCard.place(display)`

Place card on ground

**Kind:** instance method of `SageStoneCard`

Param	Type
display	String

`SorcerousSpellWallCard` - `EnvironmentCard`

`SorcerousSpellWallCard`

**Kind:** global class

**Extends:** `EnvironmentCard`

- `SorcerousSpellWallCard` - `EnvironmentCard`
  - `.applyEnvironment(monsterCard)`
  - `.place(display)`
  - `.takeSnapshot()` : `Object`
  - `.endTurn()`
  - `.canInvoke(game, player, invokeParams)` : `Boolean`
  - `.invoke(game, player, invokeParams)`
  - `.canSummon(display, pose)` : `Boolean`
  - `.summon(display, pose)`
  - `.canPlace(display)` : `Boolean`

`sorcerousSpellWallCard.applyEnvironment(monsterCard)`

apply effect to a monster card

**Kind:** instance method of `SorcerousSpellWallCard`

Param	Type
monsterCard	object

`sorcerousSpellWallCard.place(display)`

set the display of environment card

**Kind:** instance method of `SorcerousSpellWallCard`

Param	Type
display	string

**sorcerousSpellWallCard.takeSnapshot() : Object**

Take snapshot

**Kind:** instance method of [SorcerousSpellWallCard](#)

**sorcerousSpellWallCard.endTurn()**

End turn.

**Kind:** instance method of [SorcerousSpellWallCard](#)

**sorcerousSpellWallCard.canInvoke(game, player, invokeParams) : Boolean**

Check if can invoke

**Kind:** instance method of [SorcerousSpellWallCard](#)

Param	Type
game	Game
player	Player
invokeParams	Array

**sorcerousSpellWallCard.invoke(game, player, invokeParams)**

Invoke card effects

**Kind:** instance method of [SorcerousSpellWallCard](#)

Param	Type
game	Game
player	Player
invokeParams	Array

**sorcerousSpellWallCard.canSummon(display, pose) : Boolean**

Check if can summon

**Kind:** instance method of [SorcerousSpellWallCard](#)

Param	Type
display	String
pose	String

**sorcerousSpellWallCard.summon(display, pose)**

Summon

**Kind:** instance method of [SorcerousSpellWallCard](#)

Param	Type
display	String
pose	String

**sorcerousSpellWallCard.canPlace(display) : Boolean**

Check if can place

**Kind:** instance method of [SorcerousSpellWallCard](#)

Param	Type
display	String

### [SpellbookOfEternityCard](#) - [SpellCard](#)

SpellbookOfEternityCard

**Kind:** global class

**Extends:** [SpellCard](#)

- [SpellbookOfEternityCard](#) - [SpellCard](#)
  - [.canPlace\(\)](#) : Boolean
  - [.place\(display\)](#)
  - [.canInvoke\(game, player, invokeParams\)](#) : Boolean
  - [.takeSnapshot\(\)](#) : Object

#### [spellbookOfEternityCard.canPlace\(\)](#) : Boolean

Check if can place

**Kind:** instance method of [SpellbookOfEternityCard](#)

#### [spellbookOfEternityCard.place\(display\)](#)

Place card on ground

**Kind:** instance method of [SpellbookOfEternityCard](#)

Param	Type
display	String

#### [spellbookOfEternityCard.canInvoke\(game, player, invokeParams\)](#) : Boolean

Check if can invoke

**Kind:** instance method of [SpellbookOfEternityCard](#)

**Overrides:** [canInvoke](#)

Param	Type
game	Game
player	Player
invokeParams	Array

#### [spellbookOfEternityCard.takeSnapshot\(\)](#) : Object

Take snapshot

**Kind:** instance method of [SpellbookOfEternityCard](#)

### [SpellbookOfSecretsCard](#) - [SpellCard](#)

SpellbookOfSecretsCard

**Kind:** global class

**Extends:** SpellCard

- [SpellbookOfSecretsCard](#) - SpellCard
  - [.canInvoke\(game, player, invokeParams\)](#) : Boolean
  - [.invoke\(game, player, invokeParams\)](#)
  - [.canPlace\(\)](#) : Boolean
  - [.place\(display\)](#)
  - [.takeSnapshot\(\)](#) : Object

[spellbookOfSecretsCard.canInvoke\(game, player, invokeParams\)](#) : Boolean

Check if can invoke

**Kind:** instance method of [SpellbookOfSecretsCard](#)

**Overrides:** [canInvoke](#)

Param	Type
game	Game
player	Player
invokeParams	Array

[spellbookOfSecretsCard.invoke\(game, player, invokeParams\)](#)

Invoke card effects

**Kind:** instance method of [SpellbookOfSecretsCard](#)

Param	Type
game	Game
player	Player
invokeParams	Array

[spellbookOfSecretsCard.canPlace\(\)](#) : Boolean

Check if can place

**Kind:** instance method of [SpellbookOfSecretsCard](#)

[spellbookOfSecretsCard.place\(display\)](#)

Place card on ground

**Kind:** instance method of [SpellbookOfSecretsCard](#)

Param	Type
display	String

[spellbookOfSecretsCard.takeSnapshot\(\)](#) : Object

Take snapshot

**Kind:** instance method of [SpellbookOfSecretsCard](#)



## ThousandKnivesCard - SpellCard

ThousandKnivesCard

**Kind:** global class

**Extends:** SpellCard

- **ThousandKnivesCard** - SpellCard
  - .canInvoke(game, player, invokeParams) : Boolean
  - .invoke(game, player, invokeParams)
  - .takeSnapshot() : Object
  - .canPlace() : Boolean
  - .place(display)

## thousandKnivesCard.canInvoke(game, player, invokeParams) : Boolean

**Kind:** instance method of ThousandKnivesCard

**Overrides:** canInvoke

Param	Type
game	Game
player	Player
invokeParams	Array.<String>

## thousandKnivesCard.invoke(game, player, invokeParams)

**Kind:** instance method of ThousandKnivesCard

Param	Type
game	Game
player	Player
invokeParams	Array.<String>

## thousandKnivesCard.takeSnapshot() : Object

Take snapshot

**Kind:** instance method of ThousandKnivesCard

**Overrides:** takeSnapshot

## thousandKnivesCard.canPlace() : Boolean

Check if can place

**Kind:** instance method of ThousandKnivesCard

## thousandKnivesCard.place(display)

Place card on ground

**Kind:** instance method of ThousandKnivesCard

Param	Type
-------	------

display	String
---------	--------

## TwistedSpaceCard - EnvironmentCard

TwistedSpaceCard

**Kind:** global class

**Extends:** [EnvironmentCard](#)

- TwistedSpaceCard - EnvironmentCard
  - .applyEnvironment(monsterCard)
  - .place(display)
  - .takeSnapshot() : Object
  - .endTurn()
  - .canInvoke(game, player, invokeParams) : Boolean
  - .invoke(game, player, invokeParams)
  - .canSummon(display, pose) : Boolean
  - .summon(display, pose)
  - .canPlace(display) : Boolean

```
twistedSpaceCard.applyEnvironment(monsterCard)
```

apply effect to a monster card

**Kind:** instance method of `TwistedSpaceCard`

Param	Type
monsterCard	object

```
twistedSpaceCard.place(display)
```

set the display of environment card

**Kind:** instance method of `TwistedSpaceCard`

Param	Type
display	string

## twistedSpaceCard.takeSnapshot() : Object

Take snapshot

**Kind:** instance method of `TwistedSpaceCard`

```
twistedSpaceCard.endTurn()
```

End turn.

**Kind:** instance method of `TwistedSpaceCard`

`twistedSpaceCard.canInvoke(game, player, invokeParams) : Boolean`

Check if can invoke

**Kind:** instance method of `TwistedSpaceCard`

Param	Type
game	Game
player	Player
invokeParams	Array

**twistedSpaceCard.invoke(game, player, invokeParams)**

Invoke card effects

**Kind:** instance method of [TwistedSpaceCard](#)

Param	Type
game	Game
player	Player
invokeParams	Array

**twistedSpaceCard.canSummon(display, pose) : Boolean**

Check if can summon

**Kind:** instance method of [TwistedSpaceCard](#)

Param	Type
display	String
pose	String

**twistedSpaceCard.summon(display, pose)**

Summon

**Kind:** instance method of [TwistedSpaceCard](#)

Param	Type
display	String
pose	String

**twistedSpaceCard.canPlace(display) : Boolean**

Check if can place

**Kind:** instance method of [TwistedSpaceCard](#)

Param	Type
display	String

**ZaborgTheThunderMonarchCard - MonsterCard**

ZaborgTheThunderMonarchCard

**Kind:** global class

**Extends:** [MonsterCard](#)

- [ZaborgTheThunderMonarchCard - MonsterCard](#)
  - [.canInvoke\(game, player, invokeParams\) : Boolean](#)
  - [.invoke\(game, player, invokeParams\)](#)

- .takeSnapshot() : Object
- .canSummon(display, pose) : Boolean
- .summon(display, pose)
- .canChangeDisplay(display) : Boolean
- .canChangePose(pose) : Boolean
- .changeDisplay(display)
- .changePose(pose)
- .canAttack() : Boolean
- .attack()
- .endTurn()
- .canPlace(display) : Boolean
- .place(display)

**zaborgTheThunderMonarchCard.canInvoke(game, player, invokeParams) : Boolean**

Check if can invoke

**Kind:** instance method of [ZaborgTheThunderMonarchCard](#)

**Overrides:** [canInvoke](#)

Param	Type
game	Game
player	Player
invokeParams	Array

**zaborgTheThunderMonarchCard.invoke(game, player, invokeParams)**

invoke power

**Kind:** instance method of [ZaborgTheThunderMonarchCard](#)

**Overrides:** [invoke](#)

Param	Type	Description
game	Game	
player	Player	
invokeParams	Array.<String>	[0]: monsterId

**zaborgTheThunderMonarchCard.takeSnapshot() : Object**

take snapshot of the current card

**Kind:** instance method of [ZaborgTheThunderMonarchCard](#)

**Overrides:** [takeSnapshot](#)

**Returns:** Object - the snapshot of the card

**zaborgTheThunderMonarchCard.canSummon(display, pose) : Boolean**

check if can summon

**Kind:** instance method of [ZaborgTheThunderMonarchCard](#)

Param	Type
display	String
pose	String

[zaborgTheThunderMonarchCard.summon\(display, pose\)](#)

summon

**Kind:** instance method of [ZaborgTheThunderMonarchCard](#)

Param	Type
display	String
pose	String

[zaborgTheThunderMonarchCard.canChangeDisplay\(display\) : Boolean](#)

check if can change display

**Kind:** instance method of [ZaborgTheThunderMonarchCard](#)

Param	Type
display	String

[zaborgTheThunderMonarchCard.canChangePose\(pose\) : Boolean](#)

check if can change pose

**Kind:** instance method of [ZaborgTheThunderMonarchCard](#)

Param	Type
pose	String

[zaborgTheThunderMonarchCard.changeDisplay\(display\)](#)

change display

**Kind:** instance method of [ZaborgTheThunderMonarchCard](#)

Param	Type
display	String

[zaborgTheThunderMonarchCard.changePose\(pose\)](#)

change pose

**Kind:** instance method of [ZaborgTheThunderMonarchCard](#)

Param	Type
pose	String

[zaborgTheThunderMonarchCard.canAttack\(\) : Boolean](#)

check can attack

**Kind:** instance method of [ZaborgTheThunderMonarchCard](#)

[zaborgTheThunderMonarchCard.attack\(\)](#)

attack

**Kind:** instance method of [ZaborgTheThunderMonarchCard](#)

[zaborgTheThunderMonarchCard.endTurn\(\)](#)

end turn

**Kind:** instance method of [ZaborgTheThunderMonarchCard](#)

[zaborgTheThunderMonarchCard.canPlace\(display\) : Boolean](#)

Check if can place

**Kind:** instance method of [ZaborgTheThunderMonarchCard](#)

Param	Type
display	String

[zaborgTheThunderMonarchCard.place\(display\)](#)

Place card on ground

**Kind:** instance method of [ZaborgTheThunderMonarchCard](#)

Param	Type
display	String

## Network APIs

Name	Parameter	Description
cl_register	{name: String, password: String}	Register using name and password.
cl_login	{name: String, password: String}	Login using name and password.
cl_update	{newName: String, newBio: String}	Update user information with new name and new bio.
cl_store_buy_life	undefined	Buy life upgrade.
cl_get_players	undefined	Get all players.
cl_following	playerTo: String	Add following.
cl_create_room	{name: String}	Create a room.
cl_join_room	{roomId: String}	Join a room.
cl_watch_room	{roomId: String}	Watch a room.
cl_leave_room	undefined	Leave the room.
cl_room_send_message	{message: String}	Send message in the room.
cl_room_propose	undefined	Propose to start a game.
cl_room_agree	undefined	Agree to the proposal.
cl_room_start	undefined	Start the game.
cl_game_action	{name: String, params: Array<String>}	Update the game.

## Dependencies

### Backend

Name	Version	URL
chai	4.2.0	<a href="http://chaijs.com">http://chaijs.com</a>
chromedriver	2.46.0	<a href="https://github.com/giggio/node-chromedriver">https://github.com/giggio/node-chromedriver</a>
coveralls	3.0.2	<a href="https://github.com/nickmerwin/node-coveralls#readme">https://github.com/nickmerwin/node-coveralls#readme</a>
debug	4.1.1	<a href="https://github.com/visionmedia/debug#readme">https://github.com/visionmedia/debug#readme</a>
eslint	5.12.1	<a href="https://eslint.org">https://eslint.org</a>
eslint-config-google	0.11.0	<a href="https://github.com/google/eslint-config-google#readme">https://github.com/google/eslint-config-google#readme</a>
express	4.16.4	<a href="http://expressjs.com/">http://expressjs.com/</a>
http-server	0.11.1	<a href="https://github.com/indexzero/http-server#readme">https://github.com/indexzero/http-server#readme</a>
husky	1.3.1	<a href="https://github.com/typicode/husky#readme">https://github.com/typicode/husky#readme</a>
jsdoc	3.5.5	<a href="https://github.com/jsdoc3/jsdoc#readme">https://github.com/jsdoc3/jsdoc#readme</a>
mocha	5.2.0	<a href="https://mochajs.org">https://mochajs.org</a>
mongoose	5.4.9	<a href="http://mongoosejs.com">http://mongoosejs.com</a>
nyc	13.1.0	<a href="https://github.com/istanbuljs/nyc#readme">https://github.com/istanbuljs/nyc#readme</a>
selenium-webdriver	4.0.0	<a href="https://github.com/SeleniumHQ/selenium">https://github.com/SeleniumHQ/selenium</a>
socket.io	2.2.0	<a href="https://github.com/socketio/socket.io#readme">https://github.com/socketio/socket.io#readme</a>

### Frontend

Name	Version	URL
atomizer	3.5.3	<a href="https://github.com/acss-io/atomizer">https://github.com/acss-io/atomizer</a>
debug	4.1.1	<a href="https://github.com/visionmedia/debug#readme">https://github.com/visionmedia/debug#readme</a>
eslint	5.15.1	<a href="https://eslint.org">https://eslint.org</a>
eslint-config-google	0.12.0	<a href="https://github.com/google/eslint-config-google#readme">https://github.com/google/eslint-config-google#readme</a>
react	16.8.4	<a href="https://reactjs.org/">https://reactjs.org/</a>
react-dom	16.8.4	<a href="https://reactjs.org/">https://reactjs.org/</a>
react-scripts	2.1.8	<a href="https://github.com/facebook/create-react-app#readme">https://github.com/facebook/create-react-app#readme</a>
socket.io-client	2.2.0	<a href="https://github.com/Automattic/socket.io-client#readme">https://github.com/Automattic/socket.io-client#readme</a>