

---

---

# TeamTris

---

---

TEAMTRIS CODE DOCUMENTATION  
WEST LAFAYETTE, IN  
MARCH 5TH 2020

CREATED BY

STEVEN DELLAMORE  
INDHU RAMANATHAN  
RICHARD HANSEN  
COLUMBUS HOLT  
*dellamoresteven@gmail.com*  
*TeamTris*  
*CS407*

# Contents

<b>1</b>	<b>StartScreen</b>	<b>3</b>
1.1	constructor . . . . .	3
1.2	draw . . . . .	4
1.3	animateTitle . . . . .	4
1.4	drawUsernameBox . . . . .	5
1.5	drawTitle . . . . .	5
1.6	drawTokenBox . . . . .	6
1.7	mouseClickedStart . . . . .	7
1.8	drawHighScoreButtonCheckMouse . . . . .	8
1.9	drawHighScoreButton . . . . .	8
1.10	keyPressedStart . . . . .	9
<b>2</b>	<b>Player</b>	<b>9</b>
2.1	constructor . . . . .	10
2.2	setPlayerNum . . . . .	10
<b>3</b>	<b>Team</b>	<b>11</b>
3.1	constructor . . . . .	11
3.2	addPlayer . . . . .	11
<b>4</b>	<b>General</b>	<b>12</b>
4.1	mouseClicked . . . . .	12
4.2	keyPressed . . . . .	13
<b>5</b>	<b>SingleBot</b>	<b>14</b>
5.1	SingleBot . . . . .	14
5.2	getFit . . . . .	14
5.3	GetMove . . . . .	15
<b>6</b>	<b>DoubleBot</b>	<b>15</b>
<b>7</b>	<b>TripleBot</b>	<b>16</b>
<b>8</b>	<b>BotManager</b>	<b>16</b>

<b>9</b>	<b>FrontendTests</b>	<b>16</b>
9.1	CheckSame . . . . .	17
9.2	testDefaultUsername . . . . .	18
9.3	testDefaultTokenValue . . . . .	18
9.4	testCheckInitStartScreenValues . . . . .	18
9.5	testCheckTitlePosAfterTwoDraw . . . . .	19
9.6	testChangeUserUsername . . . . .	19
9.7	testChangeMaxUsername . . . . .	19
9.8	testDeleteUsername . . . . .	20
9.9	testCheckSpecialChars . . . . .	20
9.10	testHighScoreButton . . . . .	20
9.11	testCreateGameButton . . . . .	20
9.12	testJoinLobbyButton . . . . .	21
9.13	testCheckLobbyInitValues . . . . .	21
9.14	testCheckTokenIsBeingDisplayed . . . . .	21
9.15	testAddAndRemoveBotsFromLobby . . . . .	21
9.16	checkPlayCardValues . . . . .	22
9.17	integrationTest1 . . . . .	22
9.18	testGameArrayNotNull . . . . .	22
9.19	testGameScreenRotateKeyPress . . . . .	22
9.20	testGameScreenFailRotateKeyPress . . . . .	23
9.21	testGameScreenFailRotateKeyPress . . . . .	23
9.22	testFourRotate . . . . .	23
9.23	testMove . . . . .	23
9.24	testNewSquare . . . . .	24
9.25	testNumberOfPlayers . . . . .	24
9.26	testRunnerSetupStartScreen . . . . .	24

# 1 StartScreen

**Author:** Steven Dellamore

**Description:** Startscreen will build the startscreen and create all the buttons needed for the user to get into a game with their friends. The mouseClicks and the keyboard imports all all forwarded to this class when gamestate == 0

## 1.1 constructor

**Author:** Steven Dellamore

`constructor()`

**Description:** The constructor gets called when making a startscreen object. It will init all the values and set up the socket listener for the server to send things too. Here are the init values of the class variables:

```
this.TokenBoxText = "";
this.usernameBoxStroke = false;
this.usernameText = "username";
this.usernameTextTouched = false;
this.gameStateStartScreen = 0;
this.titleAnimation = [300, 500, 400, 700];
```

These variables will be updated throughout the life of start screen. `this.TokenBoxText` will init the token box to nothing, since the user has yet to do anything. the `this.usernameBoxStroke` will be set to false so the program knows if the user as tried to submit. `this.titleAnimation = [300, 500, 400, 700];` is the starting position of the title, and will fall every X frames.

**Parameters:**

**void** : constructor takes no params

**Returns:**

**StartScreen** : An object of start class class

## 1.2 draw

**Author:** Steven Dellamore

`draw()`

**Description:** This function will be ran at 60 frames a second and will call all the functions needed to draw the launch screen. The draw function will call the title functions, the highscore functions, and call the join and create button rendering/hitboxes with `Buttonloop()`. Depending on what `this.gameStateStartScreen` is evaluated to.

```
switch (this.gameStateStartScreen) {  
    case 0:  
        this.drawUsernameBox();  
        break;  
    case 1:  
        this.drawTokenBox();  
        break;  
}
```

**Parameters:**

**void :** draw takes no arguments

**Returns:**

**void :** void

## 1.3 animateTitle

**Author:** Steven Dellamore

`animateTitle()`

**Description:** Will check and add/subtract the locations of the T's falling when you go to the launch screen.

```
if (this.titleAnimation[i] > 0) {  
    this.titleAnimation[i] -= 10;  
}
```

Once `this.titleAnimation[i]`, where `i` is between `[0,4]`, is negative, the array index will no longer be decremented.

**Parameters:**

**void** : `animateTitle` takes no arguments

**Returns:**

**void**

## 1.4 drawUsernameBox

**Author:** Steven Dellamore

`drawUsernameBox()`

**Description:** This function will draw the white username box onto the screen displaying the `this.usernameText` in the center. This function will also use `this.usernameBoxStroke` to display the red outline around the username box.

**Parameters:**

**void** : `drawUsernameBox` takes no arguments

**Returns:**

**void**

## 1.5 drawTitle

**Author:** Steven Dellamore

`drawTitle()`

**Description:** This function will draw the title (Teamtris) onto the launch screen. Also, the function will be responsible for displaying the current falling location of the two T's falling at the start of the screen. We make rects based on the current location of `this.titleAnimation`.

```

let yStart;
rect(-windowWidth / 4.3, (yStart = windowHeight / 2.6) -
    this.titleAnimation[0], squareSize, squareSize)

rect(-windowWidth / 4.3, (yStart - (spaceBetweenSquares)) -
    this.titleAnimation[0], squareSize, squareSize)

fill(255, 0, 0) // fill red

rect(-windowWidth / 4.3, yStart - (2 * spaceBetweenSquares) -
    this.titleAnimation[1], squareSize, squareSize)

rect(-windowWidth / 4.3 - spaceBetweenSquares,
    yStart - (2 * spaceBetweenSquares) - this.titleAnimation[1],
    squareSize, squareSize)

rect(-windowWidth / 4.3 + spaceBetweenSquares,
    yStart - (2 * spaceBetweenSquares) - this.titleAnimation[1],
    squareSize, squareSize)

```

The important thing to note is to see the y val of the rect is being changed by 10 every frame in `function animateTitle()`.

### Parameters:

**void** : drawTitle takes no arguments

### Returns:

**void**

## 1.6 drawTokenBox

**Author:** Steven Dellamore

`drawTokenBox()`

**Description:** This function will draw the token box once the user clicks "join game". It will display the token box and the accept button. Unlike other buttons, all mouse clicks are handled.

### Parameters:

**void** : drawTokenBox takes no arguments

**Returns:**

**void**

## 1.7 mouseClickedStart

**Author:** Steven Dellamore

`mouseClickedStart()`

**Description:** This function is being called whenever `gameState = 0` AND the user clicks their mouse. First, we will check what `this.gameStateStartScreen` is. If its 0, we will check the `function ClickedLoop()` to see if the user is clicking on the join game, create game, or highscore score buttons. If the user clicks on a the create game button with a valid username we are going to send them into the lobbyscreen.

```
// Creating my lobbyscreen object
mLobbyScreen = new LobbyScreen(
    new Player(
        this.usernameText, Math.floor(Math.random() * 100), true));

gameState = 1; // Switch to lobby screen
```

We need to create a new Player, and set their ownership value to 0. We see its constructor defined here:

```
constructor(username, id, owner){
    this.username = username;
    this.id = id;
    this.owner = owner;
    this.playerNum;
}
```

We then pass this object into the lobbyscreen and switch the `gameState = 1` to move the user to the next screen.

**Parameters:**

**void** : mouseClickedStart takes no arguments

**Returns:**

**void**



## 1.8 drawHighScoreButtonCheckMouse

**Author:** Steven Dellamore

`drawHighScoreButtonCheckMouse()`

**Description:** This function is being called whenever the user clicks with gamestate of the `this.gameStateStartScreen == 0`;. This function checks if the mouse is over the highscore button and returns `true` if it is, `false` if its not.

**Parameters:**

**void :** `drawHighScoreButtonCheckMouse` takes no arugments

**Returns:**

**bool :**

true => If mouse is over score button

false => If mouse is not over score button

## 1.9 drawHighScoreButton

**Author:** Steven Dellamore

`drawHighScoreButton()`

**Description:** This function will draw the three bars in the bottom left of the screen. It will first check what `this.drawHighScoreButtonCheckMouse()` and set accordingly:

```
let fillHighScore = "white"; // default value is white
/* Checks if the mouse is over the highscore */
if (this.drawHighScoreButtonCheckMouse()) {
    /* if the mouse is over, it will change the boxes to green */
    fillHighScore = "rgb(0,255,0)";
}
```

If `this.drawHighScoreButtonCheckMouse()` returns true, then we set `fillHighScore` to `"green"`, otherwise keep it `"white"`.

**Parameters:**

**void** : takes no arguments

**Returns:**

**void** : no return

## 1.10 keyPressedStart

**Author:** Steven Dellamore

`keyPressedStart()`

**Description:** Called whenever the `General::function keyPressed()` function routes the signal to this function. a.k.a whenever `gameState == 0`. This function first checks the `this.gameStateStartScreen` like so:

```
switch(this.gameStateStartScreen) {  
    case 0:  
        // username box active  
        ...  
    case 1:  
        // token box active  
        ...  
}
```

From here, we can figure out where the user is trying to type and add the types characters accordingly.

**Parameters:**

**void** : keyPressedStart takes no arguments

**Returns:**

**void**

## 2 Player

**Author:** Steven Dellamore, Richard Hansen

**Description:** Every user will have their own object of the Player class. This

is going to be passed around to other people in the lobby. This class will tell the game screen who is who and will help identify moves.

## 2.1 constructor

**Author:** Steven Dellamore

```
constructor (username , id , owner )
```

**Description:** The constructor takes in three things, a name, id and a owner flag. It will then create an object of `Player` and init all class variables. This Class is used throughout all stages of the program.

### Parameters:

**String username** : username of the new Player

**int id** : id, [0,4], of the new player.

**boolean owner** : `true` or `false` if they are owner

### Returns:

**Player** : An object of Player class

## 2.2 setPlayerNum

**Author:** Steven Dellamore

```
setPlayerNum (num)
```

**Description:** Will set `this.playerNum` equal to `num`. This is just a helper function.

### Parameters:

**int num** : sets the `this.playerNum = num`

### Returns:

**void** : returns nothing

## 3 Team

**Author:** Steven Dellamore, Richard Hansen

**Description:** The team class will contain all the other players that are in your game, the team name and the token for your lobby. Once new players come addPlayer will be called to push a newplayer onto the playersInTeam array.

### 3.1 constructor

**Author:** Steven Dellamore

```
constructor()
```

**Description:** The constructor gets called anytime someone joins or create a game.

**Parameters:**

**void:** no parameters

**Returns:**

**Team :** A object of the class

### 3.2 addPlayer

**Author:** Steven Dellamore

```
addPlayer(player)
```

**Description:** The add player function gets called whenever a bot or a real player joins your lobby. This function will also be called to populate the lobby when you join.

**Parameters:**

**Player player:** This parameter is the new player/bot that is joining your

team.

**Returns:**

**void** : no return

## 4 General

**Author:** Steven Dellamore, Richard Hansen

**Description:** This is an abstract class that will hold mouseClicked and key-Pressed p5 functions.

### 4.1 mouseClicked

**Author:** Steven Dellamore, Richard Hansen

mouseClicked()

**Description:** Will be called whenever the user clicks on anywhere on the screen. Once called, it will go straight into a switch to decide where to route to based on the gameState

```
switch (gameState) {  
  case 0:  
    // start screens mouseClicked  
    mStartScreen.mouseClickedStart();  
    break;  
  case 1:  
    // lobby screens mouseClicked  
    mLobbyScreen.mouseClickedLobby();  
    break;  
  case 2:  
    break;  
  case 3:  
    break;  
}
```

The variables `gameState`, `mStartScreen`, `mLobbyScreen` are all defined in `sketch.js`

**Parameters:**

**void** : takes no parameters

**Returns:**

**void** : returns nothing

## 4.2 keyPressed

**Author:** Steven Dellamore, Richard Hansen

mouseClicked()

**Description:** Will be called whenever the presses a key. Once called, it will go straight into a switch to decide where to route to based on the gameState

```
switch (gameState) {  
  case 0:  
    mStartScreen.keyPressedStart();  
    break;  
  case 1:  
    mLobbyScreen.keyPressedLobby();  
    break;  
  case 2:  
    mGameScreen.keyPressedGame();  
    break;  
  case 3:  
    // mScoreScreen.keyPressedScore();  
    break;  
}
```

The variables `gameState`, `mStartScreen`, `mLobbyScreen` are all defined in `sketch.js`

**Parameters:**

**void** : takes no parameters

**Returns:**

**void** : returns nothing

## 5 SingleBot

**Author:** JavaComSci

**Description:** Single bot extends the abstract bot class defined here:

```
public abstract class Bot {  
    public abstract List<Tuple<int, int>> GetMove(  
        Board board,  
        List<Block> blocks,  
        bool allRotations = false  
    );  
}
```

The SingleBot class will be made if the player requires only one bot in their game.

### 5.1 SingleBot

**Author:** JavaComSci

SingleBot ()

**Description:** Creates a new board for the bot.

**Parameters:**

**void :** SingleBot takes no params

**Returns:**

**SingleBot :** An object of single bot class

### 5.2 getFit

**Author:** JavaComSci

List <...> getFit (Board board , Block block , int rotation)

**Description:** need desc here TODO

**Parameters:**

**Board board** : contains the the board that we want to make the move on

**Block block** : contains the block that we want to fit

**int rotation** : which roation we are trying to fit for

**Returns:**

**List<...> compatiblePieces** : information about the pieces that are compatible on the board

### 5.3 GetMove

**Author:** JavaComSci

```
public override List<...> GetMove(Board, List<Block>, bool)
```

**Description:** need desc here TODO

**Parameters:**

**int[][] board** : current enviornment

**List<Block> blocks** : contains the list of all the blocks to try to fit in this location

**Returns:**

**List<...> bestPiecePlacementOfCurrentBlock** : contains the list of the indicies of where the piece would be on the board

## 6 DoubleBot

**Author:** JavaComSci

**Description:** DoubleBot bot extends the abstract bot class defined here:

```
public abstract class Bot {  
    public abstract List<Tuple<int, int>> GetMove(  
        Board board,  
        List<Block> blocks,  
        bool allRotations = false
```



```
    );  
}
```

The DoubleBot class will be made if the player requires two bots in their game.

## 7 TripleBot

**Author:** JavaComSci

**Description:** TripleBot extends the abstract bot class defined here:

```
public abstract class Bot {  
    public abstract List<Tuple<int, int>> GetMove(  
        Board board,  
        List<Block> blocks,  
        bool allRotations = false  
    );  
}
```

The TripleBot class will be made if the player requires three bots in their game.

## 8 BotManager

**Author:** JavaComSci

**Description:** TODO

## 9 FrontendTests

**Author:** Steven Dellamore, Richard Hansen

**Description:** This is the testing doc for all the frontend tests. We decided to not go with a framework because we didnt think we needed everything the framework gives us. This framework uses the idea of dependency injection. We mock out all the p5 variables like so:

```

global.mouseY = 30;
global.LEFT_ARROW = 37;
global.RIGHT_ARROW = 39;
global.DOWN_ARROW = 40;
global.createCanvas = function (x,y) { }
global.push = function () { }
global.pop = function () { }
global.translate = function () { }
... // Keeps going

```

This allows us to control all aspects of the test and really unit test every line of code in our functions. More over, we are able to mock out other classes that are being used by the class we are trying to test like so:

```

global.buttonList = button[0];
global.Buttons = button[1];
global.Buttonloop = button[2];
global.ClickedLoop = button[3];
global.FindButtonbyID = button[4];

```

Once again, we can really drill down to the functions and have a really good understanding of what its doing and its return values.

## 9.1 CheckSame

**Author:** Steven Dellamore, Richard Hansen

```
CheckSame( string , string , string , boolean = false )
```

**Description:** Checks to see if the given and expected strings are the same. If they are not this function will return false and print what the expected was.

```

console.log(
    red, numTests++ + ". " + name + " failed should have been " +
        expect + " but was " + given);

```

If its true it will print a success message.

**Parameters:**

**string given** : real output

**string expect** : expected output  
**string name** : name of test  
**boolean debug** : **true** if you want debug statments printed

**Returns:**

**boolean** : true if given and expected match, false otherwise

## 9.2 testDefaultUsername

**Author:** Steven Dellamore

```
async function testDefaultUsername()
```

**Description:** Checks to see if the default `mStartScreen.usernameText` is **"username"**.

## 9.3 testDefaultTokenValue

**Author:** Steven Dellamore

```
async function testDefaultTokenValue()
```

**Description:** Checks to see if the default `mStartScreen.TokenBoxText` is **" "**.

## 9.4 testCheckInitStartScreenValues

**Author:** Steven Dellamore

```
async function testCheckInitStartScreenValues()
```

**Description:** Checks to see if all the other init startscreen values are correct.

```
// check usernameTextTouched is false  
CheckSame(  
    mStartScreen.usernameTextTouched,false,  
    "checkInitStartScreenValues.usernameTextTouched");  
  
// check titleAnimation [0-4] is set to the correct values
```

```

CheckSame(
    mStartScreen.titleAnimation[0],300,
    "checkInitStartScreenValues.titleAnimation[0]");
... // other indexes of titleAnimation

// Check the stroke of the box is set to false
CheckSame(
    mStartScreen.usernameBoxStroke,false,
    "checkInitStartScreenValues.usernameBoxStroke");

```

## 9.5 testCheckTitlePosAfterTwoDraw

**Author:** Steven Dellamore

```

async function testCheckTitlePosAfterTwoDraw()

```

**Description:** Run `mStartScreen.draw()` twice and check that the title pos values have been updated correctly.

## 9.6 testChangeUserUsername

**Author:** Steven Dellamore

```

async function testChangeUserUsername()

```

**Description:** TODO

## 9.7 testChangeMaxUsername

**Author:** Steven Dellamore

```

async function testChangeMaxUsername()

```

**Description:** TODO

## 9.8 testDeleteUsername

**Author:** Steven Dellamore

```
async function testDeleteUsername()
```

**Description:** TODO

## 9.9 testCheckSpecialChars

**Author:** Steven Dellamore

```
async function testCheckSpecialChars()
```

**Description:** TODO

## 9.10 testHighScoreButton

**Author:** Steven Dellamore

```
async function testHighScoreButton()
```

**Description:** TODO

## 9.11 testCreateGameButton

**Author:** Steven Dellamore

```
async function testCreateGameButton()
```

**Description:** TODO

## 9.12 testJoinLobbyButton

**Author:** Steven Dellamore

```
async function testJoinLobbyButton()
```

**Description:** TODO

## 9.13 testCheckLobbyInitValues

**Author:** Steven Dellamore

```
async function testCheckLobbyInitValues()
```

**Description:** TODO

## 9.14 testCheckTokenIsBeingDisplayed

**Author:** Steven Dellamore

```
async function testCheckTokenIsBeingDisplayed()
```

**Description:** TODO

## 9.15 testAddAndRemoveBotsFromLobby

**Author:** Steven Dellamore

```
async function testAddAndRemoveBotsFromLobby()
```

**Description:** TODO

## **9.16 checkPlayCardValues**

**Author:** Steven Dellamore

```
async function checkPlayCardValues()
```

**Description:** TODO

## **9.17 integrationTest1**

**Author:** Steven Dellamore

```
async function integrationTest1()
```

**Description:** TODO

## **9.18 testGameArrayNotNull**

**Author:** Richard Hansen

```
async function testGameArrayNotNull()
```

**Description:** TODO

## **9.19 testGameScreenRotateKeyPress**

**Author:** Richard Hansen

```
async function testGameScreenRotateKeyPress()
```

**Description:** TODO

## **9.20 testGameScreenFailRotateKeyPress**

**Author:** Richard Hansen

```
async function testGameScreenFailRotateKeyPress()
```

**Description:** TODO

## **9.21 testGameScreenFailRotateKeyPress**

**Author:** Richard Hansen

```
async function testGameScreenFailRotateKeyPress()
```

**Description:** TODO

## **9.22 testFourRotate**

**Author:** Richard Hansen

```
async function testFourRotate()
```

**Description:** TODO

## **9.23 testMove**

**Author:** Richard Hansen

```
async function testMove()
```

**Description:** TODO



## 9.24 testNewSquare

**Author:** Richard Hansen

```
async function testNewSquare()
```

**Description:** TODO

## 9.25 testNumberOfPlayers

**Author:** Richard Hansen

```
async function testNumberOfPlayers()
```

**Description:** TODO

## 9.26 testRunnerSetupStartScreen

**Author:** Steven Dellamore, Richard Hansen

```
async function testRunnerSetupStartScreen()
```

**Description:** TODO