

# Dong\_Minh\_Doxygen

---

## Class List

---

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">🔗 Bullet</a>	
<a href="#">🔗 CameraSwitch</a>	
<a href="#">🔗 EnemyAI</a>	
<a href="#">🔗 Health</a>	
<a href="#">🔗 PlayerController</a>	
<a href="#">🔗 PlayerFire</a>	

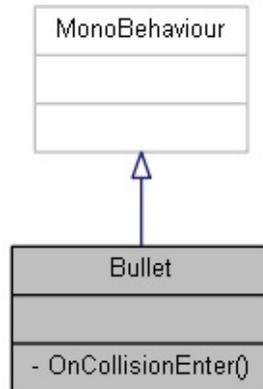
---

# Dong\_Minh\_Doxygen

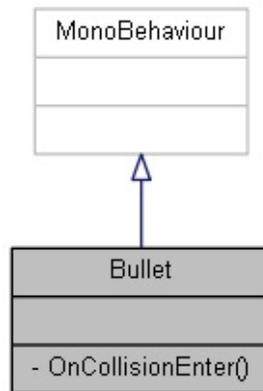
[Private Member Functions](#) | [List of all members](#)

## Bullet Class Reference

Inheritance diagram for Bullet:



Collaboration diagram for Bullet:



## Private Member Functions

---

void **OnCollisionEnter** (Collision collision)

When there is a collision detection from the enemy bullet to the player's tank. If it collides with each other, the player will take damage [More...](#)

---

## Detailed Description

---

Definition at line **5** of file **Bullet.cs**.

## Member Function Documentation

---

## ◆ OnCollisionEnter()

```
void Bullet.OnCollisionEnter ( Collision collision )
```

private

When there is a collision detection from the enemy bullet to the player's tank. If it collides with each other, the player will take damage

### Pseudo Code

If there is a detected collision between two objects

Get the health component for the tank

If it is not equal to null, player takes 1 damage

The enemy bullet object will be destroyed after

### Parameters

**hit** Determine whether there was collision to get gameObject  
**health** Get **Health** Component

### Returns

Returns if collision detected to update player's health and destroy enemy bullet

Definition at line **28** of file **Bullet.cs**.

Here is the call graph for this function:



---

The documentation for this class was generated from the following file:

- D:/Google Drive/Unity Projects/Tank Game/Assets/Scripts/**Bullet.cs**

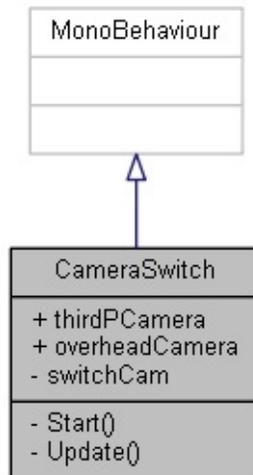
# Dong\_Minh\_Doxygen

[Public Attributes](#) | [Private Member Functions](#) |

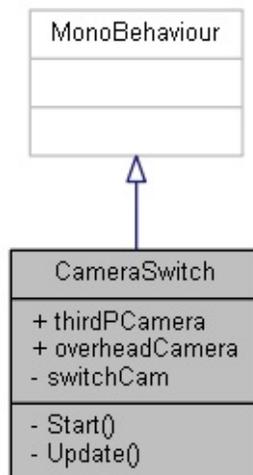
[Private Attributes](#) | [List of all members](#)

## CameraSwitch Class Reference

Inheritance diagram for CameraSwitch:



Collaboration diagram for CameraSwitch:



## Public Attributes

---

Camera **thirdPCamera**

---

Camera **overheadCamera**

---

## Private Member Functions

---

void **Start** ()

This is to initialize upon start of the game [More...](#)

void **Update** ()

This will update every frame to check whether the user switch  
cams or not. [More...](#)

---

## Private Attributes

---

bool **switchCam** = false

---

## Detailed Description

---

Definition at line **5** of file **CameraSwitch.cs**.

## Member Function Documentation

---

## ◆ Start()

```
void CameraSwitch.Start ( )
```

```
private
```

This is to initialize upon start of the game

### **Pseudo Code**

Get third person camera and set it to true

Get overhead camera and set it to false

Definition at line [22](#) of file [CameraSwitch.cs](#).

## ◆ Update()

```
void CameraSwitch.Update ( )
```

```
private
```

This will update every frame to check whether the user switch cams or not.

### Pseudo Code

Did the user press F?

If yes, check the switchCam statement, then reverse its statement, then another statement to make more statements to change the cameras

If no, do nothing.

### Parameters

**thirdPCamera** Third Person Camera

**overheadCamera** Overhead Camera

### Returns

Swaps the statement to true or false for overhead and third person camera

Definition at line [45](#) of file [CameraSwitch.cs](#).

# Member Data Documentation

---

## ◆ overheadCamera

Camera CameraSwitch.overheadCamera

Definition at line **8** of file **CameraSwitch.cs**.

---

## ◆ switchCam

```
bool CameraSwitch.switchCam = false
```

private

Definition at line **10** of file **CameraSwitch.cs**.

---

## ◆ thirdPCamera

Camera CameraSwitch.thirdPCamera

Definition at line **7** of file **CameraSwitch.cs**.

---

The documentation for this class was generated from the following file:

- D:/Google Drive/Unity Projects/Tank Game/Assets/Scripts/**CameraSwitch.cs**
- 

Generated by **doxygen** 1.8.13

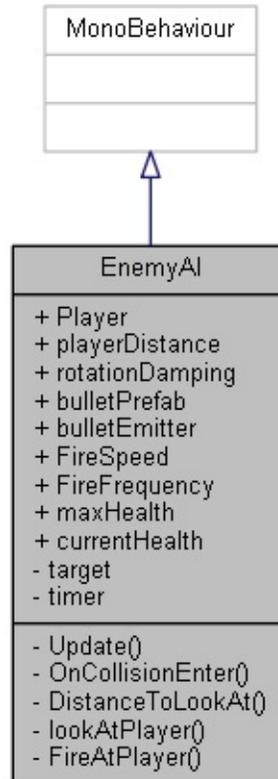
# Dong\_Minh\_Doxygen

[Public Attributes](#) | [Private Member Functions](#) |

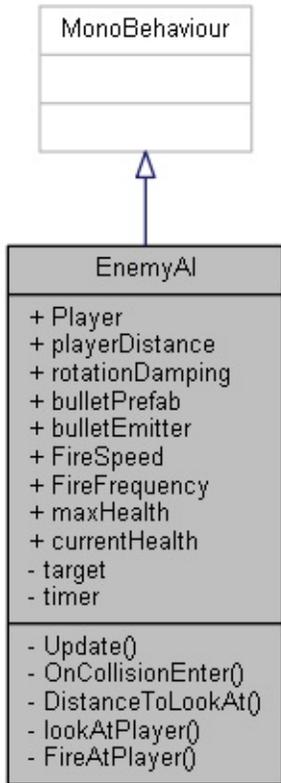
[Private Attributes](#) | [List of all members](#)

## EnemyAI Class Reference

Inheritance diagram for EnemyAI:



Collaboration diagram for EnemyAI:



## Public Attributes

---

Transform **Player**

float **playerDistance**

float **rotationDamping**

GameObject **bulletPrefab**

Transform **bulletEmitter**

float **FireSpeed**

float **FireFrequency**

const int **maxHealth = 3**

int **currentHealth = maxHealth**

---

## Private Member Functions

---

void **Update** ()

Update every per frame. Only function in here is the DistanceToLookAt. [More...](#)

void **OnCollisionEnter** (Collision collision)

When there is a collision detected by the player bullet, the enemy's life will go down. If down to 3, the enemy is dead. [More...](#)

void **DistanceToLookAt** ()

Determine a function of how far the enemy looks at the player (distance), and when to shoot at the player. [More...](#)

void **lookAtPlayer** ()

Extensive math problem here ti determine the player position and enemy position. [More...](#)

void **FireAtPlayer** ()

Fire at the player [More...](#)

---

## Private Attributes

---

RaycastHit **target**

float **timer** = 0

---

## Detailed Description

---

Definition at line **5** of file **EnemyAI.cs**.

## Member Function Documentation

---

## ◆ DistanceToLookAt()

```
void EnemyAI.DistanceToLookAt ( )
```

private

Determine a function of how far the enemy looks at the player (distance), and when to shoot at the player.

### Pseudo Code

If the player is within looking distance, I will look at him wherever the player is.

If the player steps into my danger zone, I am allowed to fire

### Parameters

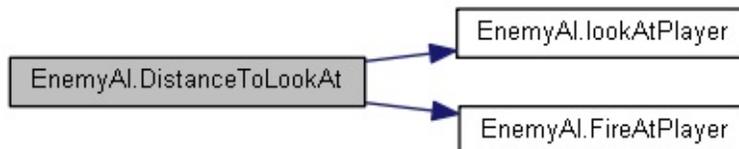
**timer** Incremental for time.deltatime  
**playerDistance** Vector3.Distance

### Returns

Returns if I'm within the distance to look at the player

Definition at line **82** of file **EnemyAI.cs**.

Here is the call graph for this function:



Here is the caller graph for this function:



## ◆ FireAtPlayer()

```
void EnemyAI.FireAtPlayer ( )
```

private

Fire at the player

### Pseudo Code

An object will be created with some velocity and will be destroyed at a certain amount of time.

#### Parameters

**BulletInstance** Create the bullet instance using the prefab stuff

**launch** Get the bullet instance/param>

#### Parameters

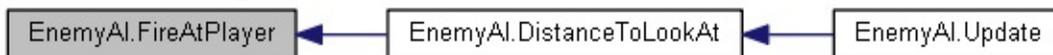
**velocity** Part of launch to determine the bullet speed

#### Returns

Fire at the player

Definition at line **128** of file **EnemyAI.cs**.

Here is the caller graph for this function:



## ◆ lookAtPlayer()

```
void EnemyAI.lookAtPlayer ( )
```

private

Extensive math problem here to determine the player position and enemy position.

### Pseudo Code

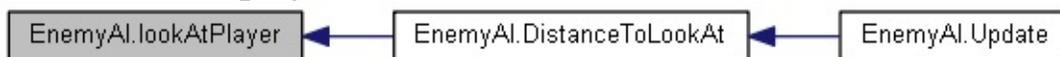
I will rotate using Quaternion to look at the player position

### Returns

Rotation

Definition at line **111** of file **EnemyAI.cs**.

Here is the caller graph for this function:



## ◆ OnCollisionEnter()

```
void EnemyAI.OnCollisionEnter ( Collision collision )
```

private

When there is a collision detected by the player bullet, the enemy's life will go down. If down to 3, the enemy is dead.

### Pseudo Code

Did I get hit by the player's bullet?

If so, I lose a hit point

Did I lose all 3 lives?

If so, destroy object

### Parameters

**currentHealth** Enemy **Health** Variable

### Returns

Return if there is something collided with enemy and player's bullet

Definition at line **55** of file **EnemyAI.cs**.

## ◆ Update()

```
void EnemyAI.Update ( )
```

private

Update every per frame. Only function in here is the DistanceToLookAt.

### Pseudo Code

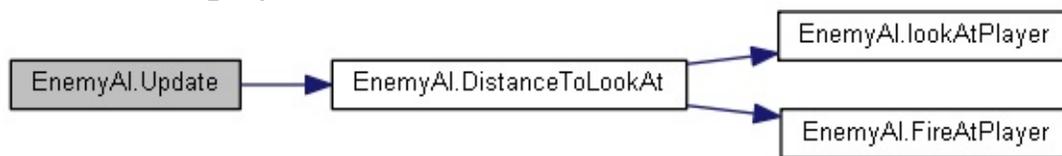
Keeps updating the DistanceToLookAt function.

### Returns

Distance Function

Definition at line **32** of file **EnemyAI.cs**.

Here is the call graph for this function:



# Member Data Documentation

---

## ◆ bulletEmitter

Transform EnemyAI.bulletEmitter

Definition at line **12** of file **EnemyAI.cs**.

---

## ◆ bulletPrefab

GameObject EnemyAI.bulletPrefab

Definition at line **11** of file **EnemyAI.cs**.

---

## ◆ currentHealth

```
int EnemyAI.currentHealth = maxHealth
```

Definition at line **20** of file **EnemyAI.cs**.

---

## ◆ FireFrequency

```
float EnemyAI.FireFrequency
```

Definition at line **14** of file **EnemyAI.cs**.

---

## ◆ FireSpeed

```
float EnemyAI.FireSpeed
```

Definition at line **13** of file **EnemyAI.cs**.

---

## ◆ maxHealth

```
const int EnemyAI.maxHealth = 3
```

Definition at line **19** of file **EnemyAI.cs**.

---

## ◆ Player

Transform EnemyAI.Player

Definition at line **7** of file **EnemyAI.cs**.

---

## ◆ playerDistance

```
float EnemyAI.playerDistance
```

Definition at line **8** of file **EnemyAI.cs**.

---

## ◆ rotationDamping

```
float EnemyAI.rotationDamping
```

Definition at line **9** of file **EnemyAI.cs**.

---

## ◆ target

RaycastHit EnemyAI.target

private

Definition at line **15** of file **EnemyAI.cs**.

---

## ◆ timer

```
float EnemyAI.timer = 0
```

private

Definition at line **17** of file **EnemyAI.cs**.

---

The documentation for this class was generated from the following file:

- D:/Google Drive/Unity Projects/Tank Game/Assets/Scripts/**EnemyAI.cs**

---

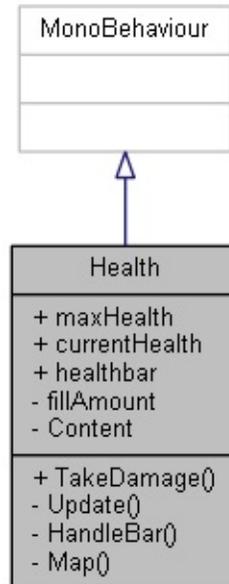
Generated by **doxygen** 1.8.13

# Dong\_Minh\_Doxygen

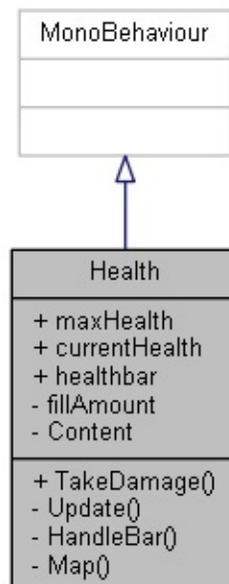
## Health Class Reference

[Public Member Functions](#) | [Public Attributes](#) |  
[Private Member Functions](#) | [Private Attributes](#) |  
[List of all members](#)

Inheritance diagram for Health:



Collaboration diagram for Health:



## Public Member Functions

---

void **TakeDamage** (int amount)

Whenever the player takes damage, the health will go down.

[More...](#)

---

## Public Attributes

---

```
const int maxHealth = 20
```

```
int currentHealth = maxHealth
```

```
RectTransform healthbar
```

---

## Private Member Functions

---

void **Update** ()

Update once per frame to update the HandleBar [More...](#)

---

void **HandleBar** ()

Will get the fill amount for the health bar to determine its mapping and will update accordingly [More...](#)

---

float **Map** (float value, float inMin, float inMax, float outMin, float outMax)

Calculations to determine the 0 to 1 Unity asset and will calculate if it is greater than 1. It will scale it down between 0 to 1. [More...](#)

---

## Private Attributes

---

float **fillAmount**

Image **Content**

---

## Detailed Description

---

Definition at line **6** of file **Health.cs**.

## Member Function Documentation

---

## ◆ HandleBar()

```
void Health.HandleBar ( )
```

```
private
```

Will get the fill amount for the health bar to determine its mapping and will update accordingly

### Returns

Healthbar's fill amount

Definition at line **35** of file **Health.cs**.

Here is the call graph for this function:



Here is the caller graph for this function:



## ◆ Map()

```
float Health.Map ( float value,  
                  float inMin,  
                  float inMax,  
                  float outMin,  
                  float outMax  
                  )
```

private

Calculations to determine the 0 to 1 Unity asset and will calculate if it is greater than 1. It will scale it down between 0 to 1.

### Returns

Returns the Map

Definition at line [46](#) of file [Health.cs](#).

Here is the caller graph for this function:



## ◆ TakeDamage()

```
void Health.TakeDamage ( int amount )
```

Whenever the player takes damage, the health will go down.

### Pseudo Code

Did I get hit, subtract one health. If I lose all my health, reload the scene

### Returns

Returns player's health

Definition at line **61** of file **Health.cs**.

Here is the caller graph for this function:



## ◆ Update()

```
void Health.Update ( )
```

```
private
```

Update once per frame to update the HandleBar

Definition at line **24** of file **Health.cs**.

Here is the call graph for this function:



# Member Data Documentation

---

## ◆ Content

Image Health.Content

private

Definition at line **17** of file **Health.cs**.

---

## ◆ currentHealth

```
int Health.currentHealth = maxHealth
```

Definition at line **9** of file **Health.cs**.

---

## ◆ fillAmount

float Health.fillAmount

private

Definition at line **14** of file **Health.cs**.

---

## ◆ healthbar

RectTransform Health.healthbar

Definition at line **10** of file **Health.cs**.

---

## ◆ maxHealth

```
const int Health.maxHealth = 20
```

Definition at line **8** of file **Health.cs**.

---

The documentation for this class was generated from the following file:

- D:/Google Drive/Unity Projects/Tank Game/Assets/Scripts/**Health.cs**
- 

Generated by doxygen 1.8.13

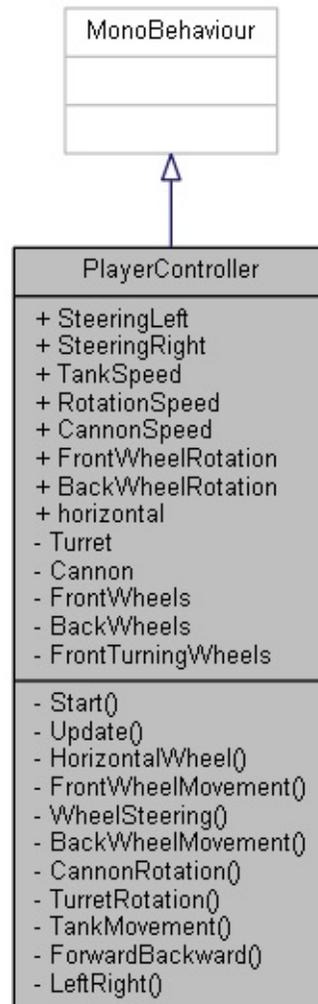
# Dong\_Minh\_Doxygen

[Public Attributes](#) | [Private Member Functions](#) |

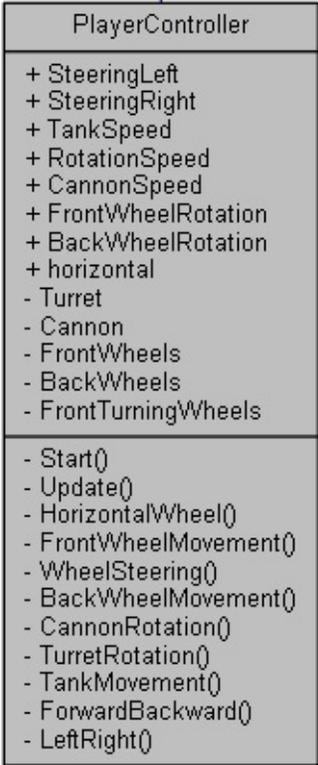
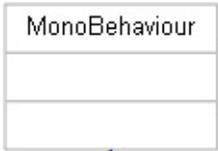
[Private Attributes](#) | [List of all members](#)

## PlayerController Class Reference

Inheritance diagram for PlayerController:



Collaboration diagram for PlayerController:



## Public Attributes

---

Transform **SteeringLeft**

Transform **SteeringRight**

float **TankSpeed**

float **RotationSpeed**

float **CannonSpeed**

float **FrontWheelRotation**

float **BackWheelRotation**

float **horizontal**

---

## Private Member Functions

---

void **Start** ()  
Initialize at the beginning of the game [More...](#)

void **Update** ()  
Update is called once per frame [More...](#)

void **HorizontalWheel** ()  
The front wheels will go either rotate left or right by 45 degrees.  
[More...](#)

void **FrontWheelMovement** ()  
The Front wheels should rotate forward/backward and should rotate with an angle of maximum of 45 to left and right [More...](#)

void **WheelSteering** ()  
The steering for left or right to determine how far it should also rotate. [More...](#)

void **BackWheelMovement** ()  
The back wheels should rotate and also accelerate/decelerate the tank [More...](#)

void **CannonRotation** ()  
How far the cannon should rotate up and to its rest position. Max for up position is vertical (90 degrees) [More...](#)

void **TurretRotation** ()  
Will rotate the turret left or right when the user press left or right arrow keys [More...](#)

void **TankMovement** ()  
This is called in the update function. All functionality is done here for the tank movement like moving forward, backwards, left, or right. [More...](#)

---

void **ForwardBackward** (float movement)

When the W or S key is pressed, the tank will move forward or backward respectively [More...](#)

---

void **LeftRight** (float movement)

When the A or D key is pressed, the tank will move left or right respectively [More...](#)

---

## Private Attributes

---

GameObject **Turret**

GameObject **Cannon**

GameObject [] **FrontWheels**

GameObject [] **BackWheels**

GameObject [] **FrontTurningWheels**

---

## Detailed Description

---

Definition at line **6** of file **PlayerController.cs**.

## Member Function Documentation

---

## ◆ BackWheelMovement()

```
void PlayerController.BackWheelMovement ( )
```

```
private
```

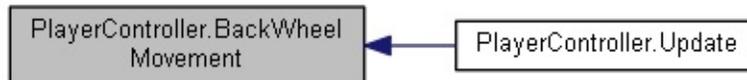
The back wheels should rotate and also accelerate/decelerate the tank

### Returns

Returns Back Wheels Rotation

Definition at line **143** of file **PlayerController.cs**.

Here is the caller graph for this function:



## ◆ CannonRotation()

```
void PlayerController.CannonRotation ( )
```

private

How far the cannon should rotate up and to its rest position. Max for up position is vertical (90 degrees)

### Returns

Returns Cannon Rotation

Definition at line **176** of file **PlayerController.cs**.

Here is the caller graph for this function:



## ◆ ForwardBackward()

```
void PlayerController.ForwardBackward ( float movement ) private
```

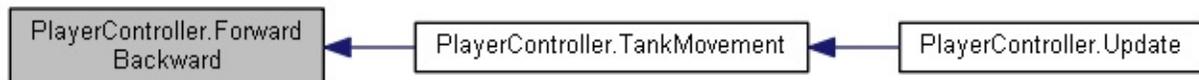
When the W or S key is pressed, the tank will move forward or backward respectively

### Returns

Returns translate

Definition at line **315** of file **PlayerController.cs**.

Here is the caller graph for this function:



## ◆ FrontWheelMovement()

```
void PlayerController.FrontWheelMovement ( )
```

```
private
```

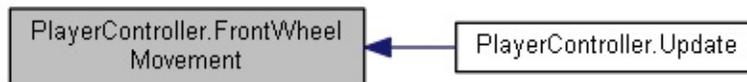
The Front wheels should rotate forward/backward and should rotate with an angle of maximum of 45 to left and right

### Returns

Returns FrontWheel rotation/

Definition at line **95** of file **PlayerController.cs**.

Here is the caller graph for this function:



## ◆ HorizontalWheel()

```
void PlayerController.HorizontalWheel ( )
```

private

The front wheels will go either rotate left or right by 45 degrees.

### Pseudo Code

If the input key is D, rotate the front wheels by 45 degrees  
else if the input key is A, rotate by -45 degrees

### Returns

Returns horizontal

Definition at line **78** of file **PlayerController.cs**.

Here is the caller graph for this function:



## ◆ LeftRight()

```
void PlayerController.LeftRight ( float movement )
```

private

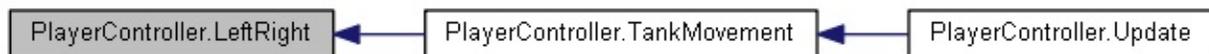
When the A or D key is pressed, the tank will move left or right respectively

### Returns

Returns rotate

Definition at line **326** of file **PlayerController.cs**.

Here is the caller graph for this function:



## ◆ Start()

```
void PlayerController.Start ( )
```

```
private
```

Initialize at the beginning of the game

Definition at line **34** of file **PlayerController.cs**.

---

## ◆ TankMovement()

```
void PlayerController.TankMovement ( )
```

private

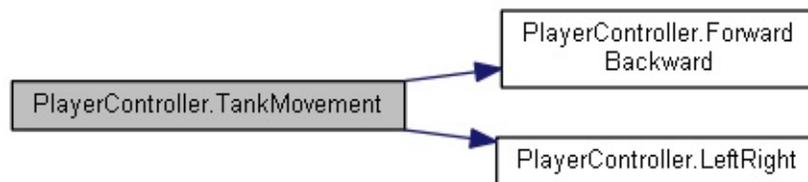
This is called in the update function. All functionality is done here for the tank movement like moving forward, backwards, left, or right.

### Returns

Returns translate or rotation

Definition at line **269** of file **PlayerController.cs**.

Here is the call graph for this function:



Here is the caller graph for this function:



## ◆ TurretRotation()

```
void PlayerController.TurretRotation ( )
```

private

Will rotate the turret left or right when the user press left or right arrow keys

### Returns

Returns Turret Rotation

Definition at line [237](#) of file [PlayerController.cs](#).

Here is the caller graph for this function:



## ◆ Update()

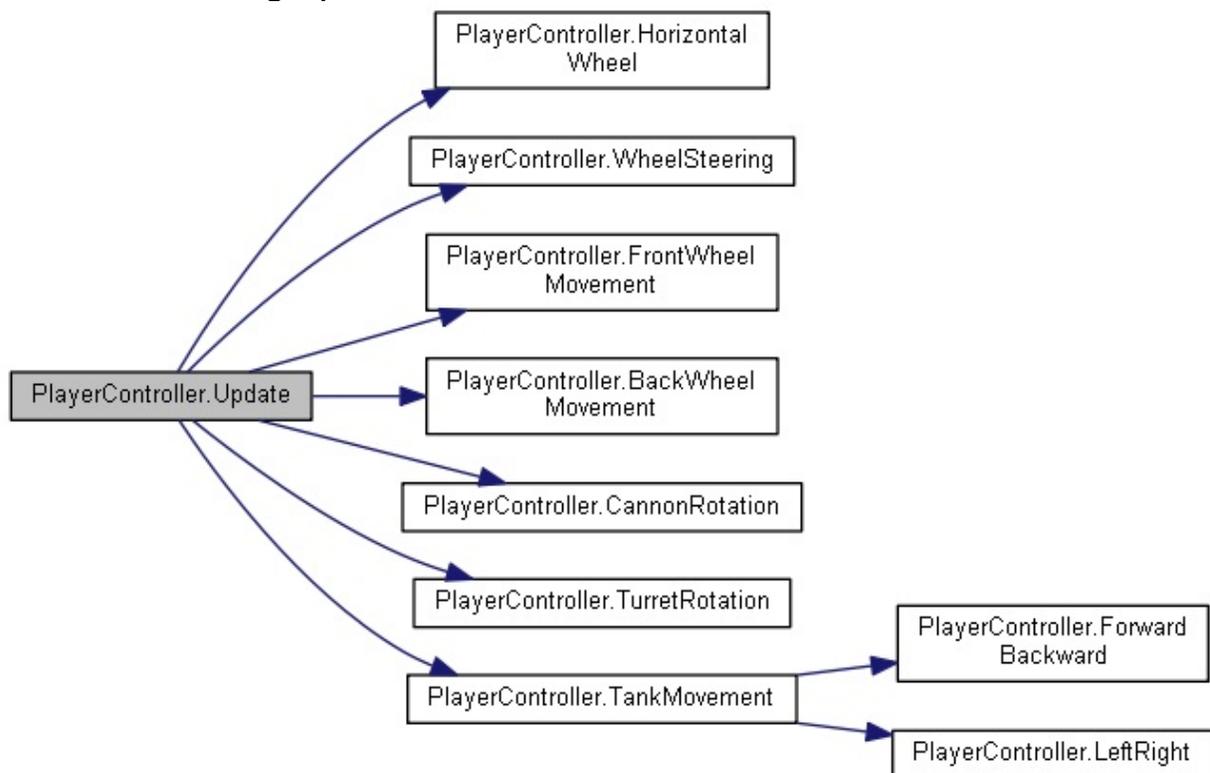
```
void PlayerController.Update ( )
```

private

Update is called once per frame

Definition at line **47** of file **PlayerController.cs**.

Here is the call graph for this function:



## ◆ WheelSteering()

```
void PlayerController.WheelSteering ( )
```

private

The steering for left or right to determine how far it should also rotate.

### Returns

Returns Steering Left or Right rotation

Definition at line **125** of file **PlayerController.cs**.

Here is the caller graph for this function:



# Member Data Documentation

---

## ◆ BackWheelRotation

```
float PlayerController.BackWheelRotation
```

Definition at line **26** of file **PlayerController.cs**.

---

## ◆ BackWheels

GameObject [] PlayerController.BackWheels

private

Definition at line **14** of file **PlayerController.cs**.

---

## ◆ Cannon

GameObject PlayerController.Cannon

private

Definition at line **12** of file **PlayerController.cs**.

---

## ◆ CannonSpeed

```
float PlayerController.CannonSpeed
```

Definition at line **24** of file **PlayerController.cs**.

---

## ◆ FrontTurningWheels

GameObject [] PlayerController.FrontTurningWheels

private

Definition at line **15** of file **PlayerController.cs**.

---

## ◆ FrontWheelRotation

```
float PlayerController.FrontWheelRotation
```

Definition at line **25** of file **PlayerController.cs**.

---

## ◆ FrontWheels

GameObject [] PlayerController.FrontWheels

private

Definition at line **13** of file **PlayerController.cs**.

---

## ◆ horizontal

```
float PlayerController.horizontal
```

Definition at line **27** of file **PlayerController.cs**.

---

## ◆ RotationSpeed

```
float PlayerController.RotationSpeed
```

Definition at line **23** of file **PlayerController.cs**.

---

## ◆ SteeringLeft

Transform PlayerController.SteeringLeft

Definition at line **16** of file **PlayerController.cs**.

---

## ◆ SteeringRight

Transform PlayerController.SteeringRight

Definition at line **17** of file **PlayerController.cs**.

---

## ◆ TankSpeed

```
float PlayerController.TankSpeed
```

Definition at line **22** of file **PlayerController.cs**.

---

## ◆ Turret

GameObject PlayerController.Turret

private

Definition at line **11** of file **PlayerController.cs**.

---

The documentation for this class was generated from the following file:

- D:/Google Drive/Unity Projects/Tank Game/Assets/Scripts/**PlayerController.cs**

---

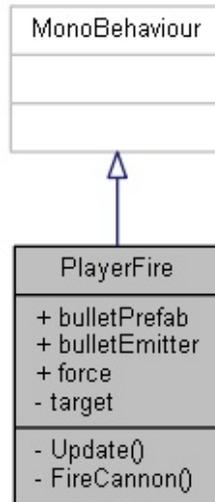
Generated by **doxygen** 1.8.13

# Dong\_Minh\_Doxygen

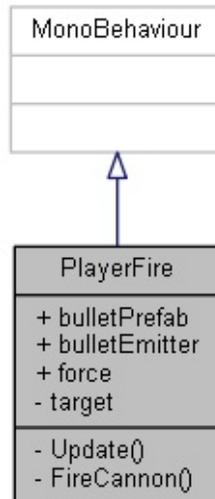
[Public Attributes](#) | [Private Member Functions](#) |  
[Private Attributes](#) | [List of all members](#)

## PlayerFire Class Reference

Inheritance diagram for PlayerFire:



Collaboration diagram for PlayerFire:



## Public Attributes

---

GameObject **bulletPrefab**

Transform **bulletEmitter**

float **force = 100f**

---

## Private Member Functions

---

void **Update** ()

Update once per frame [More...](#)

void **FireCannon** ()

When the player presses the spacebar, fire the cannon. [More...](#)

---

## Private Attributes

---

RaycastHit **target**

---

## Detailed Description

---

Definition at line **4** of file **PlayerFire.cs**.

## Member Function Documentation

---

## ◆ FireCannon()

```
void PlayerFire.FireCannon ( )
```

private

When the player presses the spacebar, fire the cannon.

### Pseudo Code

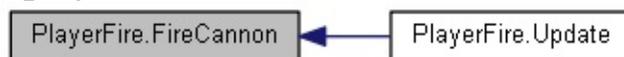
Did I press the spacebar? If so, create the bullet, add some velocity, and fire!

### Returns

Returns gameobject

Definition at line **31** of file **PlayerFire.cs**.

Here is the caller graph for this function:



## ◆ Update()

```
void PlayerFire.Update ( )
```

```
private
```

Update once per frame

Definition at line **16** of file **PlayerFire.cs**.

Here is the call graph for this function:



# Member Data Documentation

---

## ◆ bulletEmitter

Transform PlayerFire.bulletEmitter

Definition at line **7** of file **PlayerFire.cs**.

---

## ◆ bulletPrefab

GameObject PlayerFire.bulletPrefab

Definition at line **6** of file **PlayerFire.cs**.

---

## ◆ force

```
float PlayerFire.force = 100f
```

Definition at line **8** of file **PlayerFire.cs**.

---

## ◆ target

RaycastHit PlayerFire.target

private

Definition at line **9** of file **PlayerFire.cs**.

---

The documentation for this class was generated from the following file:

- D:/Google Drive/Unity Projects/Tank Game/Assets/Scripts/**PlayerFire.cs**

---

Generated by **doxygen** 1.8.13

# Dong\_Minh\_Doxygen

---

## Class Index

---

[b](#) | [c](#) | [e](#) | [h](#) | [p](#)

**b**

Bullet

**c**

CameraSwitch

**e**

EnemyAI

**h**

Health

**p**

PlayerController

PlayerFire

[b](#) | [c](#) | [e](#) | [h](#) | [p](#)

---

Generated by [doxygen](#) 1.8.13

# Dong\_Minh\_Doxygen

---

## Class Hierarchy

---

[Go to the graphical class hierarchy](#)

This inheritance list is sorted roughly, but not completely, alphabetically:

[detail level 1 2]

▼  **MonoBehaviour**

 **Bullet**

 **CameraSwitch**

 **EnemyAI**

 **Health**

 **PlayerController**

 **PlayerFire**

---

# Dong\_Minh\_Doxygen

---

Here is a list of all class members with links to the classes they belong to:

## - b -

- BackWheelMovement() : [PlayerController](#)
- BackWheelRotation : [PlayerController](#)
- BackWheels : [PlayerController](#)
- bulletEmitter : [EnemyAI](#) , [PlayerFire](#)
- bulletPrefab : [EnemyAI](#) , [PlayerFire](#)

## - c -

- Cannon : [PlayerController](#)
- CannonRotation() : [PlayerController](#)
- CannonSpeed : [PlayerController](#)
- Content : [Health](#)
- currentHealth : [EnemyAI](#) , [Health](#)

## - d -

- DistanceToLookAt() : [EnemyAI](#)

## - f -

- fillAmount : [Health](#)
- FireAtPlayer() : [EnemyAI](#)
- FireCannon() : [PlayerFire](#)
- FireFrequency : [EnemyAI](#)
- FireSpeed : [EnemyAI](#)
- force : [PlayerFire](#)
- ForwardBackward() : [PlayerController](#)
- FrontTurningWheels : [PlayerController](#)
- FrontWheelMovement() : [PlayerController](#)
- FrontWheelRotation : [PlayerController](#)
- FrontWheels : [PlayerController](#)

## - h -

- HandleBar() : **Health**
- healthbar : **Health**
- horizontal : **PlayerController**
- HorizontalWheel() : **PlayerController**

## - l -

- LeftRight() : **PlayerController**
- lookAtPlayer() : **EnemyAI**

## - m -

- Map() : **Health**
- maxHealth : **EnemyAI , Health**

## - o -

- OnCollisionEnter() : **Bullet , EnemyAI**
- overheadCamera : **CameraSwitch**

## - p -

- Player : **EnemyAI**
- playerDistance : **EnemyAI**

## - r -

- rotationDamping : **EnemyAI**
- RotationSpeed : **PlayerController**

## - s -

- Start() : **CameraSwitch , PlayerController**
- SteeringLeft : **PlayerController**
- SteeringRight : **PlayerController**
- switchCam : **CameraSwitch**

**- t -**

- TakeDamage() : **Health**
- TankMovement() : **PlayerController**
- TankSpeed : **PlayerController**
- target : **EnemyAI** , **PlayerFire**
- thirdPCamera : **CameraSwitch**
- timer : **EnemyAI**
- Turret : **PlayerController**
- TurretRotation() : **PlayerController**

**- u -**

- Update() : **CameraSwitch** , **EnemyAI** , **Health** , **PlayerController** , **PlayerFire**

**- w -**

- WheelSteering() : **PlayerController**

# Dong\_Minh\_Doxygen

---

- BackWheelMovement() : **PlayerController**
- CannonRotation() : **PlayerController**
- DistanceToLookAt() : **EnemyAI**
- FireAtPlayer() : **EnemyAI**
- FireCannon() : **PlayerFire**
- ForwardBackward() : **PlayerController**
- FrontWheelMovement() : **PlayerController**
- HandleBar() : **Health**
- HorizontalWheel() : **PlayerController**
- LeftRight() : **PlayerController**
- lookAtPlayer() : **EnemyAI**
- Map() : **Health**
- OnCollisionEnter() : **Bullet** , **EnemyAI**
- Start() : **CameraSwitch** , **PlayerController**
- TakeDamage() : **Health**
- TankMovement() : **PlayerController**
- TurretRotation() : **PlayerController**
- Update() : **CameraSwitch** , **EnemyAI** , **Health** , **PlayerController** , **PlayerFire**
- WheelSteering() : **PlayerController**

# Dong\_Minh\_Doxygen

---

## - b -

- BackWheelRotation : **PlayerController**
- BackWheels : **PlayerController**
- bulletEmitter : **EnemyAI , PlayerFire**
- bulletPrefab : **EnemyAI , PlayerFire**

## - c -

- Cannon : **PlayerController**
- CannonSpeed : **PlayerController**
- Content : **Health**
- currentHealth : **EnemyAI , Health**

## - f -

- fillAmount : **Health**
- FireFrequency : **EnemyAI**
- FireSpeed : **EnemyAI**
- force : **PlayerFire**
- FrontTurningWheels : **PlayerController**
- FrontWheelRotation : **PlayerController**
- FrontWheels : **PlayerController**

## - h -

- healthbar : **Health**
- horizontal : **PlayerController**

## - m -

- maxHealth : **EnemyAI , Health**

## - o -

- overheadCamera : **CameraSwitch**

**- p -**

- Player : **EnemyAI**
- playerDistance : **EnemyAI**

**- r -**

- rotationDamping : **EnemyAI**
- RotationSpeed : **PlayerController**

**- s -**

- SteeringLeft : **PlayerController**
- SteeringRight : **PlayerController**
- switchCam : **CameraSwitch**

**- t -**

- TankSpeed : **PlayerController**
- target : **EnemyAI** , **PlayerFire**
- thirdPCamera : **CameraSwitch**
- timer : **EnemyAI**
- Turret : **PlayerController**

# Dong\_Minh\_Doxygen

---

## File List

---

Here is a list of all files with brief descriptions:

[detail level 1 2 3]

▼ <b>Assets</b>	
▼ <b>Scripts</b>	
<b>Bullet.cs</b>	
<b>CameraSwitch.cs</b>	
<b>EnemyAI.cs</b>	
<b>Health.cs</b>	
<b>PlayerController.cs</b>	
<b>PlayerFire.cs</b>	

---

Generated by **doxygen** 1.8.13

# Dong\_Minh\_Doxygen

Assets

## Assets Directory Reference

# Directories

---

directory **Scripts**

---

# Dong\_Minh\_Doxygen

Assets > Scripts >

## Scripts Directory Reference

## Files

---

file [Bullet.cs](#) [code]

file [CameraSwitch.cs](#) [code]

file [EnemyAI.cs](#) [code]

file [Health.cs](#) [code]

file [PlayerController.cs](#) [code]

file [PlayerFire.cs](#) [code]

---

# Dong\_Minh\_Doxygen

Assets > Scripts >

Classes

## **Bullet.cs File Reference**

[Go to the source code of this file.](#)

# Classes

---

class **Bullet**

---

---

Generated by doxygen 1.8.13

# Dong\_Minh\_Doxygen

Assets > Scripts >

Classes

## CameraSwitch.cs File Reference

[Go to the source code of this file.](#)

# Classes

---

class **CameraSwitch**

---

---

Generated by doxygen 1.8.13

# Dong\_Minh\_Doxygen

Assets > Scripts >

Classes

## EnemyAI.cs File Reference

[Go to the source code of this file.](#)

# Classes

---

class **EnemyAI**

---

---

Generated by doxygen 1.8.13

# Dong\_Minh\_Doxygen

Assets > Scripts >

Classes

## Health.cs File Reference

[Go to the source code of this file.](#)

# Classes

---

class **Health**

---

---

Generated by doxygen 1.8.13

# Dong\_Minh\_Doxygen

Assets > Scripts >

Classes

## **PlayerController.cs** **File Reference**

[Go to the source code of this file.](#)

# Classes

---

class **PlayerController**

---

---

Generated by doxygen 1.8.13

# Dong\_Minh\_Doxygen

Assets > Scripts >

Classes

## PlayerFire.cs File Reference

[Go to the source code of this file.](#)

# Classes

---

class **PlayerFire**

---

---

Generated by doxygen 1.8.13

# Dong\_Minh\_Doxygen

---

## Bullet Member List

---

This is the complete list of members for **Bullet**, including all inherited members.

**OnCollisionEnter**(Collision collision) **Bullet** private

---

Generated by doxygen 1.8.13

# Dong\_Minh\_Doxygen

Assets > Scripts

## Bullet.cs

[Go to the documentation of this file.](#)

```
1 using UnityEngine;
2 using System.Collections;
3 using System.Collections.Generic;
4
5 public class Bullet : MonoBehaviour
6 {
28     void OnCollisionEnter(Collision collision)
29     {
30         var hit = collision.gameObject;
31         var health = hit.GetComponent<Health>
    ();
32         if (health != null)
33         {
34             health.TakeDamage(1);
35         }
36
37         if(GameObject.FindGameObjectWithTag("tankObject"
    ))
38             Destroy(gameObject);
39     }
40 }
```

# Dong\_Minh\_Doxygen

---

## CameraSwitch Member List

---

This is the complete list of members for **CameraSwitch**, including all inherited members.

<b>overheadCamera</b>	<b>CameraSwitch</b>	
<b>Start()</b>	<b>CameraSwitch</b>	private
<b>switchCam</b>	<b>CameraSwitch</b>	private
<b>thirdPCamera</b>	<b>CameraSwitch</b>	
<b>Update()</b>	<b>CameraSwitch</b>	private

# Dong\_Minh\_Doxygen

Assets > Scripts

## CameraSwitch.cs

[Go to the documentation of this file.](#)

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class CameraSwitch : MonoBehaviour {
6
7     public Camera thirdPCamera;
8     public Camera overheadCamera;
9
10    private bool switchCam = false;
11
12    void Start () {
13
14        thirdPCamera.GetComponent<Camera>
15        ().enabled = true;
16        overheadCamera.GetComponent<Camera>
17        ().enabled = false;
18    }
19
20    void Update () {
21
22        // get the input key
23        if (Input.GetKeyDown ("f")) {
24            switchCam = !switchCam;
25        }
26
27        // check if the switch camera happened
28        if (switchCam == true) {
29
30        }
31    }
32
33    }
34
```

```
55 |         thirdPCamera.GetComponent<Camera>  
    |         ().enabled = false;  
56 |  
    |         overheadCamera.GetComponent<Camera> ().enabled =  
    |         true;  
57 |     }  
58 |     else {  
59 |         thirdPCamera.GetComponent<Camera>  
    |         ().enabled = true;  
60 |  
    |         overheadCamera.GetComponent<Camera> ().enabled =  
    |         false;  
61 |     }  
62 | }  
63 | }
```

# Dong\_Minh\_Doxygen

---

## EnemyAI Member List

---

This is the complete list of members for **EnemyAI**, including all inherited members.

<b>bulletEmitter</b>	<b>EnemyAI</b>	
<b>bulletPrefab</b>	<b>EnemyAI</b>	
<b>currentHealth</b>	<b>EnemyAI</b>	
<b>DistanceToLookAt()</b>	<b>EnemyAI</b>	private
<b>FireAtPlayer()</b>	<b>EnemyAI</b>	private
<b>FireFrequency</b>	<b>EnemyAI</b>	
<b>FireSpeed</b>	<b>EnemyAI</b>	
<b>lookAtPlayer()</b>	<b>EnemyAI</b>	private
<b>maxHealth</b>	<b>EnemyAI</b>	
<b>OnCollisionEnter(Collision collision)</b>	<b>EnemyAI</b>	private
<b>Player</b>	<b>EnemyAI</b>	
<b>playerDistance</b>	<b>EnemyAI</b>	
<b>rotationDamping</b>	<b>EnemyAI</b>	
<b>target</b>	<b>EnemyAI</b>	private
<b>timer</b>	<b>EnemyAI</b>	private
<b>Update()</b>	<b>EnemyAI</b>	private

# Dong\_Minh\_Doxygen

Assets > Scripts >

## EnemyAI.cs

[Go to the documentation of this file.](#)

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class EnemyAI : MonoBehaviour {
6
7     public Transform Player; // prefab for the
    player
8     public float playerDistance; // will be
    measured later to check for distance
9     public float rotationDamping; // how
    dampened the rotation should be
10
11     public GameObject bulletPrefab;
12     public Transform bulletEmitter;
13     public float FireSpeed; // how fast the
    bullet should travel?
14     public float FireFrequency; // how often
    they will fire?
15     RaycastHit target;
16
17     private float timer = 0;
18
19     public const int maxHealth = 3;
20     public int currentHealth = maxHealth;
21
22     void Update () {
23         DistanceToLookAt ();
24     }
```

```

35
55     void OnCollisionEnter (Collision
      collision){
56
57         if (GameObject.FindGameObjectWithTag
      ("playerBullet")) {
58             Destroy
      (collision.collider.gameObject);
59             currentHealth -= 1;
60         }
61
62         if (currentHealth == 0)
63             Destroy(gameObject);
64     }
65
82     void DistanceToLookAt(){
83         playerDistance = Vector3.Distance
      (Player.position, transform.position);
84
85         timer += Time.deltaTime;
86
87         if (playerDistance < 150f) {
88             lookAtPlayer ();
89             Debug.Log ("Looking at player!");
90         }
91
92         if (playerDistance < 75f) {
93             if (timer > FireFrequency) {
94                 FireAtPlayer ();
95                 timer = 0;
96             }
97             Debug.Log ("Firing at the
      player!");
98         }
99     }
100
111     void lookAtPlayer(){

```

```
112 |         Quaternion rotation =
      |         Quaternion.LookRotation (Player.position -
      |         transform.position);
113 |         transform.rotation = Quaternion.Slerp
      |         (transform.rotation, rotation, Time.deltaTime *
      |         rotationDamping);
114 |     }
115 |
128 |     void FireAtPlayer(){
129 |         GameObject BulletInstance =
      |         Instantiate(bulletPrefab,
      |         bulletEmitter.position, bulletEmitter.rotation)
      |         as GameObject;
130 |         Rigidbody launch =
      |         BulletInstance.GetComponent<Rigidbody>();
131 |         launch.velocity = FireSpeed *
      |         bulletEmitter.forward;
132 |         Destroy (BulletInstance, 6f);
133 |     }
134 | }
```

# Dong\_Minh\_Doxygen

---

## Health Member List

---

This is the complete list of members for **Health**, including all inherited members.

<b>Content</b>	<b>Health</b>
<b>currentHealth</b>	<b>Health</b>
<b>fillAmount</b>	<b>Health</b>
<b>HandleBar()</b>	<b>Health</b>
<b>healthbar</b>	<b>Health</b>
<b>Map(float value, float inMin, float inMax, float outMin, float outMax)</b>	<b>Health</b>
<b>maxHealth</b>	<b>Health</b>
<b>TakeDamage(int amount)</b>	<b>Health</b>
<b>Update()</b>	<b>Health</b>

---

# Dong\_Minh\_Doxygen

Assets > Scripts

## Health.cs

[Go to the documentation of this file.](#)

```
1 using UnityEngine;
2 using UnityEngine.UI;
3 using System.Collections;
4 using UnityEngine.SceneManagement;
5
6 public class Health : MonoBehaviour
7 {
8     public const int maxHealth = 20;
9     public int currentHealth = maxHealth;
10    public RectTransform healthbar;
11
12
13    [SerializeField]
14    private float fillAmount;
15
16    [SerializeField]
17    private Image Content;
18
24    void Update(){
25        HandleBar ();
26    }
27
35    private void HandleBar(){
36        Content.fillAmount =
    Map(currentHealth, 0, maxHealth, 0, 1);
37    }
38
46    private float Map(float value, float
    inMin, float inMax, float outMin, float outMax)
```

```

47 |     {
48 |         return (value - inMin) * (outMax -
    | outMin) / (inMax - inMin) + outMin;
49 |     }
50 |
61 |     public void TakeDamage(int amount)
62 |     {
63 |         currentHealth -= amount;
64 |         if (currentHealth <= 0)
65 |         {
66 |             currentHealth = 0;
67 |             Debug.Log("Dead!");
68 |
    | //Destroy(GameObject.FindGameObjectWithTag("tank
    | Object"));
69 |             SceneManager.LoadScene
    | (SceneManager.GetActiveScene ().name);
70 |         }
71 |
72 |         //healthbar.sizeDelta = new Vector2
    | (currentHealth, healthbar.sizeDelta.y);
73 |     } // end take damage function
74 |
75 |
76 | }

```

# Dong\_Minh\_Doxygen

## PlayerController Member List

This is the complete list of members for **PlayerController**, including all inherited members.

<b>BackWheelMovement()</b>	<b>PlayerController</b>	private
<b>BackWheelRotation</b>	<b>PlayerController</b>	
<b>BackWheels</b>	<b>PlayerController</b>	private
<b>Cannon</b>	<b>PlayerController</b>	private
<b>CannonRotation()</b>	<b>PlayerController</b>	private
<b>CannonSpeed</b>	<b>PlayerController</b>	
<b>ForwardBackward(float movement)</b>	<b>PlayerController</b>	private
<b>FrontTurningWheels</b>	<b>PlayerController</b>	private
<b>FrontWheelMovement()</b>	<b>PlayerController</b>	private
<b>FrontWheelRotation</b>	<b>PlayerController</b>	
<b>FrontWheels</b>	<b>PlayerController</b>	private
<b>horizontal</b>	<b>PlayerController</b>	
<b>HorizontalWheel()</b>	<b>PlayerController</b>	private
<b>LeftRight(float movement)</b>	<b>PlayerController</b>	private
<b>RotationSpeed</b>	<b>PlayerController</b>	
<b>Start()</b>	<b>PlayerController</b>	private
<b>SteeringLeft</b>	<b>PlayerController</b>	
<b>SteeringRight</b>	<b>PlayerController</b>	
<b>TankMovement()</b>	<b>PlayerController</b>	private
<b>TankSpeed</b>	<b>PlayerController</b>	
<b>Turret</b>	<b>PlayerController</b>	private
<b>TurretRotation()</b>	<b>PlayerController</b>	private
<b>Update()</b>	<b>PlayerController</b>	private

WheelSteering()

PlayerController private

---

Generated by doxygen 1.8.13

# Dong\_Minh\_Doxygen

Assets > Scripts

## PlayerController.cs

[Go to the documentation of this file.](#)

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class PlayerController : MonoBehaviour
7 {
8     // ----- \\
9     // Game Objects \\
10    // ----- \\
11    GameObject Turret;
12    GameObject Cannon;
13    GameObject[] FrontWheels;
14    GameObject[] BackWheels;
15    GameObject[] FrontTurningWheels;
16    public Transform SteeringLeft;
17    public Transform SteeringRight;
18
19    // ----- \\
20    // Public Variables for use in Unity \\
21    // ----- \\
22    public float TankSpeed; // how fast the
    tank will move
23    public float RotationSpeed; // how fast
    the tank can rotate
24    public float CannonSpeed; // how fast the
    cannon can rotate
25    public float FrontWheelRotation; // how
```

```

    fast the front wheels rotate
26 |     public float BackWheelRotation; // how
    fast the back wheels rotate
27 |     public float horizontal;
28 |
34 |     void Start () {
35 |         Turret =
    GameObject.FindGameObjectWithTag
    ("TurretObject");
36 |         Cannon =
    GameObject.FindGameObjectWithTag
    ("CannonObject");
37 |         FrontWheels =
    GameObject.FindGameObjectsWithTag
    ("FrontWheelsObject");
38 |         BackWheels =
    GameObject.FindGameObjectsWithTag
    ("BackWheelsObject");
39 |         FrontTurningWheels =
    GameObject.FindGameObjectsWithTag
    ("FrontWheelsTurningObject");
40 |     } // end start
41 |
47 |     void Update () {
48 |         // stuff that works
49 |         HorizontalWheel ();
50 |         WheelSteering ();
51 |         FrontWheelMovement ();
52 |         BackWheelMovement ();
53 |         CannonRotation ();
54 |         TurretRotation ();
55 |         TankMovement ();
56 |         if (Input.GetKeyDown (KeyCode.R)) {
57 |             SceneManager.LoadScene
    (SceneManager.GetActiveScene ().name);
58 |         }
59 |     } // end void update

```

```

60
61
62     // ----- \\
63     //     Wheel MOVEMENT SECTION     \\
64     // ----- \\
65
78 void HorizontalWheel(){
79     // fill this later
80     if (Input.GetKey(KeyCode.D))
81     {
82         horizontal =
Input.GetKey(KeyCode.D) ? 0.45f : 0;
83     }
84     else
85         horizontal =
Input.GetKey(KeyCode.A) ? -0.45f : 0;
86     }
87
95 void FrontWheelMovement(){
96
97     // For moving forward or backward
98     if (Input.GetKey (KeyCode.W)) {
99
100         //Debug.Log ("W Key is pressed");
101
102         var RotateForward = Time.deltaTime
* FrontWheelRotation;
103
104         FrontWheels[0].transform.Rotate(RotateForward,
0, 0);
105         FrontWheels[1].transform.Rotate(RotateForward,
0, 0);
106     } // end get key for W
107     else if (Input.GetKey (KeyCode.S)) {
108

```

```

109         //Debug.Log ("S Key is pressed");
110
111         var RotateForward = Time.deltaTime
    * FrontWheelRotation;
112
113         FrontWheels[0].transform.Rotate(-
    RotateForward, 0, 0);
114         FrontWheels[1].transform.Rotate(-
    RotateForward, 0, 0);
115     } // end get key for S
116 } // end front wheel movement function
117
125 void WheelSteering(){
126     float rot = 100;
127     float current_rot = rot * horizontal;
128
129     Vector3 front_rotationAmount =
    transform.rotation.eulerAngles;
130     front_rotationAmount.y += current_rot;
131
132     SteeringLeft.rotation =
    Quaternion.Euler(front_rotationAmount);
133     SteeringRight.rotation =
    Quaternion.Euler(front_rotationAmount);
134 }
135
143 void BackWheelMovement(){
144     if (Input.GetKey (KeyCode.W)) {
145
146         //Debug.Log ("W Key is pressed");
147
148         var RotateBackward =
    Time.deltaTime * BackWheelRotation;
149
150     BackWheels[0].transform.Rotate(RotateBackward,
    0, 0);

```

```

151 | BackWheels[1].transform.Rotate(RotateBackward,
    | 0, 0);
152 |     } // end get key for W
153 |     else if (Input.GetKey (KeyCode.S)) {
154 |
155 |         //Debug.Log ("S Key is pressed");
156 |
157 |         var RotateBackward =
    | Time.deltaTime * BackWheelRotation;
158 |
159 |         BackWheels[0].transform.Rotate(-
    | RotateBackward, 0, 0);
160 |         BackWheels[1].transform.Rotate(-
    | RotateBackward, 0, 0);
161 |     } // end get key for S
162 | }
163 |
164 |
165 | // ----- \\
166 | //   CANNON MOVEMENT SECTION   \\
167 | // ----- \\
168 |
176 | void CannonRotation(){
177 |
178 |     var tilt = Time.deltaTime *
    | CannonSpeed;
179 |
180 |     if (Input.GetKey (KeyCode.UpArrow)) {
181 |         Debug.Log ("Up Arrow Key
    | Pressed");
182 |
183 |         float currentRotationZ =
    | Cannon.transform.rotation.z * 100;
184 |
185 |         //Debug.Log (currentRotationZ);
186 |

```

```

187         if (currentRotationZ <= -70){
188
189             //Debug.Log ("Input Up key is
being called for first if");
190
191             currentRotationZ = -70 / 100;
192         }
193         else {
194             //Debug.Log ("Input Up key is
being called for second if");
195
196             currentRotationZ /= 100;
197             currentRotationZ -= tilt;
198         }
199
200         Cannon.transform.Rotate (0, 0,
currentRotationZ);
201     }
202     else if (Input.GetKey
(KeyCode.DownArrow)) {
203         Debug.Log ("Down Arrow Key
Pressed");
204
205         float currentRotationZ =
Cannon.transform.rotation.z * 100;
206
207         //Debug.Log (currentRotationZ);
208
209         if (currentRotationZ >= 0){
210
211             //Debug.Log ("Input Down key
is being called for first if");
212
213             currentRotationZ = 0 / 100;
214         }
215         else{
216             //Debug.Log ("Input Down key

```

```

    is being called for second if");
217
218         currentRotationZ /= 100;
219         currentRotationZ -= tilt;
220     }
221
222         Cannon.transform.Rotate (0, 0, -
currentRotationZ);
223     }
224 } // end cannon rotation function
225
226 // ----- \\
227 //   TURRET MOVEMENT SECTION   \\
228 // ----- \\
229
237 void TurretRotation(){
238
239     if (Input.GetKey (KeyCode.LeftArrow))
    {
240         Debug.Log ("Left Arrow Key
    Pressed");
241
242         var rotateLeft = Time.deltaTime *
    RotationSpeed;
243
244         Turret.transform.Rotate (0, -
    rotateLeft, 0);
245         //Cannon.transform.Rotate (0, -
    rotateLeft, 0);
246     }
247     else if (Input.GetKey
    (KeyCode.RightArrow)) {
248         Debug.Log ("Right Arrow Key
    Pressed");
249
250         var rotateRight = Time.deltaTime *
    RotationSpeed;

```

```

251
252         Turret.transform.Rotate (0,
rotateRight, 0);
253         //Cannon.transform.Rotate (0,
rotateRight, 0);
254     }
255
256     } // end turret rotation function
257
258     // ----- \\
259     //     TANK MOVEMENT SECTION     \\
260     // ----- \\
261
262 private void TankMovement(){
263     // W or D keys
264     if (Input.GetKey (KeyCode.W)) {
265
266         Debug.Log ("W Key is pressed");
267
268         var MoveForward = Time.deltaTime *
TankSpeed;
269
270         ForwardBackward (-MoveForward);
271
272     } // end get key for W
273     else if (Input.GetKey (KeyCode.S)) {
274
275         Debug.Log ("S Key is pressed");
276
277         var MoveBackward = Time.deltaTime
* TankSpeed;
278
279         ForwardBackward (MoveBackward);
280     }
281
282     // A or D keys
283     if (Input.GetKey (KeyCode.A)) {

```

```

291
292         Debug.Log ("A Key is pressed");
293
294         var TurnLeft = Time.deltaTime *
    TankSpeed;
295
296         LeftRight (-TurnLeft);
297     }
298     else if (Input.GetKey (KeyCode.D)) {
299
300         Debug.Log ("D Key is pressed");
301
302         var TurnRight = Time.deltaTime *
    TankSpeed;
303
304         LeftRight (TurnRight);
305     }
306 }
307
315     private void ForwardBackward (float
movement){
316         transform.Translate (movement, 0, 0);
317     } // end ForwardBackward function
318
326     private void LeftRight (float movement){
327         transform.Rotate (0, movement, 0);
328     } // end LeftRight function
329 }

```

# Dong\_Minh\_Doxygen

---

## PlayerFire Member List

---

This is the complete list of members for **PlayerFire**, including all inherited members.

<b>bulletEmitter</b>	<b>PlayerFire</b>	
<b>bulletPrefab</b>	<b>PlayerFire</b>	
<b>FireCannon()</b>	<b>PlayerFire</b>	private
<b>force</b>	<b>PlayerFire</b>	
<b>target</b>	<b>PlayerFire</b>	private
<b>Update()</b>	<b>PlayerFire</b>	private

---

# Dong\_Minh\_Doxygen

Assets > Scripts

## PlayerFire.cs

[Go to the documentation of this file.](#)

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class PlayerFire : MonoBehaviour
5 {
6     public GameObject bulletPrefab;
7     public Transform bulletEmitter;
8     public float force = 100f;
9     RaycastHit target;
10
16 void Update ()
17 {
18     FireCannon ();
19 }
20
31 void FireCannon()
32 {
33     if (Input.GetKeyDown("space")) {
34         GameObject BulletInstance =
        Instantiate(bulletPrefab,
        bulletEmitter.position, bulletEmitter.rotation)
        as GameObject;
35         Rigidbody launch =
        BulletInstance.GetComponent<Rigidbody>();
36         launch.velocity = force *
        bulletEmitter.forward;
37         Destroy (BulletInstance, 5f);
38     }
39 }
```



# Dong\_Minh\_Doxygen

---

## Class Hierarchy

---

[Go to the textual class hierarchy](#)

